(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2021-0135249
(43) 공개일자 2021년11월12일

(51) 국제특허분류(Int. Cl.)
H04N 19/13 (2014.01) *H04N 19/176* (2014.01)
H04N 19/184 (2014.01) *H04N 19/577* (2014.01)
H04N 19/70 (2014.01)
(52) CPC특허분류
H04N 19/13 (2015.01)
H04N 19/176 (2015.01)
(21) 출원번호 10-2021-7029007
(22) 출원일자(국제) 2020년03월04일
심사청구일자 없음
(85) 번역문제출일자 2021년09월09일
(86) 국제출원번호 PCT/US2020/021011
(87) 국제공개번호 WO 2020/185468
국제공개일자 2020년09월17일
(30) 우선권주장
19305278.4 2019년03월11일
유럽특허청(EPO)(EP)
19305693.4 2019년05월29일
유럽특허청(EPO)(EP)

(71) 출원인
인터디지털 브이씨 홀딩스 인코포레이티드
미국 19809 델라웨어주 윌밍턴 스위트 300 벨뷰 파크웨이 200
(72) 발명자
르레아넥, 파브리쓰
프랑스 35576 쎄쏭 쎄비네 쎄에스 17616 자크 데샹 블랑 아브 데 샹 블랑 975
푸아리에, 땅기
프랑스 35576 쎄쏭 쎄비네 쎄에스 17616 자크 데샹 블랑 아브 데 샹 블랑 975
천, 야
프랑스 35576 쎄쏭 쎄비네 쎄에스 17616 자크 데샹 블랑 아브 데 샹 블랑 975
(74) 대리인
양영준, 김연송, 백만기

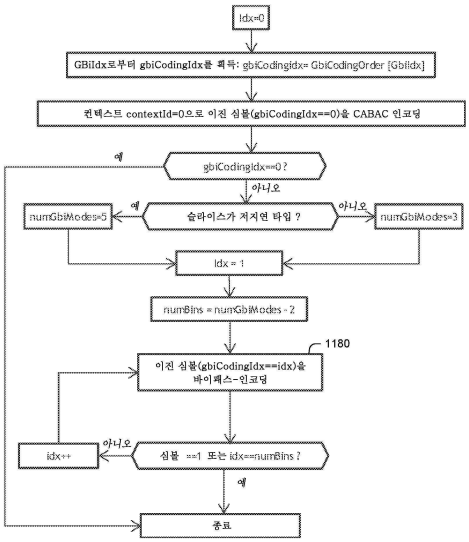
전체 청구항 수 : 총 28 항

(54) 발명의 명칭 **정규 코딩된 빈들의 수의 감소**

(57) 요 약

CABAC로 인코딩하기 위해, 비-이진 신택스 요소 값은 이진화 프로세스를 통해 이진 시퀀스(빈 스트링)에 맵핑된다. 이진 산술 코딩 엔진은 정규(컨텍스트 기반) 모드 및 바이패스 모드에서 빈을 인코딩 또는 디코딩한다.

대 표 도 - 도11a



(52) CPC특허분류
H04N 19/184 (2015.01)
H04N 19/577 (2015.01)
H04N 19/70 (2015.01)

명 세 서

청구범위

청구항 1

방법으로서,

비트스트림으로부터 복수의 이진 심볼들을 디코딩하는 단계 – 상기 복수의 이진 심볼들 중 제1 이진 심볼은 컨텍스트 기반 모드를 이용하여 엔트로피 디코딩되고, 상기 제1 이진 심볼에 후속하는 각각의 이진 심볼은 바이패스 모드에서 엔트로피 디코딩됨 –;

이진화 방식에 대응하는, 상기 복수의 이진 심볼들에 의해 표현된 인덱스를 획득하는 단계; 및

2개의 예측기들의 가중 합으로서 블록에 대한 예측을 형성하는 단계

를 포함하고, 상기 인덱스는 상기 가중 합을 형성할 때 상기 2개의 예측기들을 가중 시에 이용된 개개의 가중 인자들을 표시하는, 방법.

청구항 2

제1항에 있어서,

상기 이진화 방식은 절단 라이스 이진화 방식인, 방법.

청구항 3

제1항 또는 제2항에 있어서,

상기 인덱스는, 가중 인자들의 세트로부터, 상기 가중 합을 형성할 때 상기 2개의 예측기들 중 하나를 가중하기 위한 가중 인자를 표시하는, 방법.

청구항 4

제3항에 있어서,

다른 가중 인자가 상기 가중 인자에 응답하여 획득되고, 상기 다른 가중 인자는 상기 가중 합을 형성할 때 상기 2개의 예측기들 중 다른 하나를 가중 시에 이용되는, 방법.

청구항 5

제1항 내지 제4항 중 어느 한 항에 있어서,

상기 제1 이진 심볼은 상기 가중 합을 형성할 때 상기 2개의 예측기들에 동일한 가중치가 적용되는지를 표시하는, 방법.

청구항 6

제1항 내지 제5항 중 어느 한 항에 있어서,

상기 복수의 이진 심볼들 중 제2 이진 심볼은 상기 가중 합을 형성할 때 상기 2개의 예측기들 중 하나에 5의 가중 인자가 적용되는지를 표시하는, 방법.

청구항 7

제1항 내지 제6항 중 어느 한 항에 있어서,

상기 복수의 이진 심볼들 중 제3 이진 심볼은 상기 가중 합을 형성할 때 상기 2개의 예측기들 중 하나에 3의 가중 인자가 적용되는지를 표시하는, 방법.

청구항 8

방법으로서,

인코딩될 블록에 액세스하는 단계;

2개의 예측기들의 가중 합으로서 상기 블록에 대한 예측을 형성하는 단계; 및

상기 가중 합을 형성할 때 상기 2개의 예측기들을 가중 시에 이용된 개개의 가중 인자들을 표시하기 위한 인덱스를 인코딩하는 단계

를 포함하고,

상기 인덱스는 이진화 방식을 이용하여 복수의 이진 심볼들로 이진화되고,

상기 복수의 이진 심볼들 중 제1 이진 심볼은 컨텍스트 기반 모드를 이용하여 엔트로피 인코딩되고,

상기 제1 이진 심볼에 후속하는 각각의 이진 심볼은 바이패스 모드에서 엔트로피 인코딩되는, 방법.

청구항 9

제8항에 있어서,

상기 이진화 방식은 절단 라이스 이진화 방식인, 방법.

청구항 10

제8항 또는 제9항에 있어서,

상기 인덱스는, 가중 인자들의 세트로부터, 상기 가중 합을 형성할 때 상기 2개의 예측기들 중 하나를 가중하기 위한 가중 인자를 표시하는, 방법.

청구항 11

제10항에 있어서,

다른 가중 인자가 상기 가중 인자에 응답하여 획득되고, 상기 다른 가중 인자는 상기 가중 합을 형성할 때 상기 2개의 예측기들 중 다른 하나를 가중 시에 이용되는, 방법.

청구항 12

제8항 내지 제11항 중 어느 한 항에 있어서,

상기 제1 이진 심볼은 상기 가중 합을 형성할 때 상기 2개의 예측기들에 동일한 가중치가 적용되는지를 표시하는, 방법.

청구항 13

제8항 내지 제12항 중 어느 한 항에 있어서,

상기 복수의 이진 심볼들 중 제2 이진 심볼은 상기 가중 합을 형성할 때 상기 2개의 예측기들 중 하나에 5의 가중 인자가 적용되는지를 표시하는, 방법.

청구항 14

제8항 내지 제13항 중 어느 한 항에 있어서,

상기 복수의 이진 심볼들 중 제3 이진 심볼은 상기 가중 합을 형성할 때 상기 2개의 예측기들 중 하나에 3의 가중 인자가 적용되는지를 표시하는, 방법.

청구항 15

하나 이상의 프로세서를 포함하는 장치로서,

상기 하나 이상의 프로세서는,

비트스트림으로부터 복수의 이진 심볼들을 디코딩하고 – 상기 복수의 이진 심볼들 중 제1 이진 심볼은 컨텍스

트 기반 모드를 이용하여 엔트로피 디코딩되고, 상기 제1 이진 심볼에 후속하는 각각의 이진 심볼은 바이패스 모드에서 엔트로피 디코딩됨 -;

이진화 방식에 대응하는, 상기 복수의 이진 심볼들에 의해 표현된 인덱스를 획득하고;

2개의 예측기들의 가중 합으로서 블록에 대한 예측을 형성하도록

구성되고, 상기 인덱스는 상기 가중 합을 형성할 때 상기 2개의 예측기들을 가중 시에 이용된 개개의 가중 인자들을 표시하는, 장치.

청구항 16

제15항에 있어서,

상기 이진화 방식은 절단 라이스 이진화 방식인, 장치.

청구항 17

제15항 또는 제16항에 있어서,

상기 인덱스는, 가중 인자들의 세트로부터, 상기 가중 합을 형성할 때 상기 2개의 예측기들 중 하나를 가중하기 위한 가중 인자를 표시하는, 장치.

청구항 18

제17항에 있어서,

다른 가중 인자가 상기 가중 인자에 응답하여 획득되고, 상기 다른 가중 인자는 상기 가중 합을 형성할 때 상기 2개의 예측기들 중 다른 하나를 가중 시에 이용되는, 장치.

청구항 19

제15항 내지 제18항 중 어느 한 항에 있어서,

상기 제1 이진 심볼은 상기 가중 합을 형성할 때 상기 2개의 예측기들에 동일한 가중치가 적용되는지를 표시하는, 장치.

청구항 20

제15항 내지 제19항 중 어느 한 항에 있어서,

상기 복수의 이진 심볼들 중 제2 이진 심볼은 상기 가중 합을 형성할 때 상기 2개의 예측기들 중 하나에 5의 가중 인자가 적용되는지를 표시하는, 장치.

청구항 21

제15항 내지 제20항 중 어느 한 항에 있어서,

상기 복수의 이진 심볼들 중 제3 이진 심볼은 상기 가중 합을 형성할 때 상기 2개의 예측기들 중 하나에 3의 가중 인자가 적용되는지를 표시하는, 장치.

청구항 22

하나 이상의 프로세서를 포함하는 장치로서,

상기 하나 이상의 프로세서는,

인코딩될 블록에 액세스하고;

2개의 예측기들의 가중 합으로서 상기 블록에 대한 예측을 형성하고;

상기 가중 합을 형성할 때 상기 2개의 예측기들을 가중 시에 이용된 개개의 가중 인자들을 표시하기 위한 인덱스를 인코딩하도록 구성되고,

상기 인덱스는 이진화 방식을 이용하여 복수의 이진 심볼들로 이진화되고,

상기 복수의 이진 심볼들 중 제1 이진 심볼은 컨텍스트 기반 모드를 이용하여 엔트로피 인코딩되고,

상기 제1 이진 심볼에 후속하는 각각의 이진 심볼은 바이패스 모드에서 엔트로피 인코딩되는, 장치.

청구항 23

제22항에 있어서,

상기 이진화 방식은 절단 라이스 이진화 방식인, 장치.

청구항 24

제22항 또는 제23항에 있어서,

상기 인덱스는, 가중 인자들의 세트로부터, 상기 가중 합을 형성할 때 상기 2개의 예측기들 중 하나를 가중하기 위한 가중 인자를 표시하는, 장치.

청구항 25

제24항에 있어서,

다른 가중 인자가 상기 가중 인자에 응답하여 획득되고, 상기 다른 가중 인자는 상기 가중 합을 형성할 때 상기 2개의 예측기들 중 다른 하나를 가중 시에 이용되는, 장치.

청구항 26

제22항 내지 제25항 중 어느 한 항에 있어서,

상기 제1 이진 심볼은 상기 가중 합을 형성할 때 상기 2개의 예측기들에 동일한 가중치가 적용되는지를 표시하는, 장치.

청구항 27

제22항 내지 제26항 중 어느 한 항에 있어서,

상기 복수의 이진 심볼들 중 제2 이진 심볼은 상기 가중 합을 형성할 때 상기 2개의 예측기들 중 하나에 5의 가중 인자가 적용되는지를 표시하는, 장치.

청구항 28

제22항 내지 제27항 중 어느 한 항에 있어서,

상기 복수의 이진 심볼들 중 제3 이진 심볼은 상기 가중 합을 형성할 때 상기 2개의 예측기들 중 하나에 3의 가중 인자가 적용되는지를 표시하는, 장치.

발명의 설명

기 술 분 야

[0001] 본 실시예들은 일반적으로 비디오 인코딩 또는 디코딩을 위한 방법 및 장치에 관한 것으로서, 더 구체적으로는, 엔트로피 인코딩 및 디코딩(entropy encoding and decoding)에서 정규 코딩된 빈들(regular coded bins)의 수를 감소시키기 위한 방법 및 장치에 관한 것이다.

배 경 기 술

[0002] 높은 압축 효율을 달성하기 위해, 이미지 및 비디오 코딩 방식들은 비디오 콘텐츠에서의 공간적 및 시간적 중복성(spatial and temporal redundancy)를 레버리지(leverage)하기 위해 통상적으로 예측 및 변환을 이용한다. 일반적으로, 인트라(intra) 또는 인터(inter) 예측은 인트라 또는 인터 픽처 상관을 활용하기 위해 이용되고, 이후 종종 예측 에러들 또는 예측 잔차들로서 표시되는, 원래 블록과 예측된 블록 사이의 차이들은 변환되고, 양자화되고, 엔트로피 코딩된다. 비디오를 재구성하기 위해, 압축된 데이터는 엔트로피 코딩, 양자화, 변환 및 예측에 대응하는 역 프로세스들에 의해 디코딩된다.

발명의 내용

[0003] 실시예에 따르면, 비디오 디코딩의 방법으로서, 비트스트림으로부터 복수의 이진 심볼들을 디코딩하는 단계 – 상기 복수의 이진 심볼들 중 제1 이진 심볼은 컨텍스트 기반 모드(context-based mode)를 이용하여 엔트로피 디코딩되고, 상기 제1 이진 심볼에 후속하는 각각의 이진 심볼은 바이패스 모드(bypass mode)에서 엔트로피 디코딩됨 –; 이진화(binarization) 방식에 대응하는, 상기 복수의 이진 심볼들에 의해 표현된 인덱스를 획득하는 단계; 및 2개의 예측기들의 가중 합(weighted sum)으로서 블록에 대한 예측을 형성하는 단계 – 상기 인덱스는 상기 가중 합을 형성할 때 상기 2개의 예측기들을 가중 시에 이용된 개개의 가중 인자들을 표시함 – 를 포함하는 방법이 제공된다.

[0004] 실시예에 따르면, 비디오 인코딩의 방법으로서, 인코딩될 블록에 액세스하는 단계; 2개의 예측기들의 가중 합으로서 상기 블록에 대한 예측을 형성하는 단계; 및 상기 가중 합을 형성할 때 상기 2개의 예측기들을 가중 시에 이용된 개개의 가중 인자들을 표시하기 위한 인덱스를 인코딩하는 단계를 포함하고, 상기 인덱스는 이진화 방식을 이용하여 복수의 이진 심볼들로 이진화되고, 상기 복수의 이진 심볼들 중 제1 이진 심볼은 컨텍스트 기반 모드를 이용하여 엔트로피 인코딩되고, 상기 제1 이진 심볼에 후속하는 각각의 이진 심볼은 바이패스 모드에서 엔트로피 인코딩되는 방법이 제공된다.

[0005] 다른 실시예에 따르면, 하나 이상의 프로세서를 포함하는 비디오 디코딩을 위한 장치로서, 상기 하나 이상의 프로세서는, 비트스트림으로부터 복수의 이진 심볼들을 디코딩하고 – 상기 복수의 이진 심볼들 중 제1 이진 심볼은 컨텍스트 기반 모드를 이용하여 엔트로피 디코딩되고, 상기 제1 이진 심볼에 후속하는 각각의 이진 심볼은 바이패스 모드에서 엔트로피 디코딩됨 –; 이진화 방식에 대응하는, 상기 복수의 이진 심볼들에 의해 표현된 인덱스를 획득하고; 2개의 예측기들의 가중 합으로서 블록에 대한 예측을 형성 – 상기 인덱스는 상기 가중 합을 형성할 때 상기 2개의 예측기들을 가중 시에 이용된 개개의 가중 인자들을 표시함 – 하도록 구성되는 장치가 제공된다.

[0006] 다른 실시예에 따르면, 하나 이상의 프로세서를 포함하는 비디오 인코딩을 위한 장치로서, 상기 하나 이상의 프로세서는, 인코딩될 블록에 액세스하고; 2개의 예측기들의 가중 합으로서 상기 블록에 대한 예측을 형성하고; 상기 가중 합을 형성할 때 상기 2개의 예측기들을 가중 시에 이용된 개개의 가중 인자들을 표시하기 위한 인덱스를 인코딩하도록 구성되고, 상기 인덱스는 이진화 방식을 이용하여 복수의 이진 심볼들로 이진화되고, 상기 복수의 이진 심볼들 중 제1 이진 심볼은 컨텍스트 기반 모드를 이용하여 엔트로피 인코딩되고, 상기 제1 이진 심볼에 후속하는 각각의 이진 심볼은 바이패스 모드에서 엔트로피 인코딩되는 장치가 제공된다.

[0007] 다른 실시예에 따르면, 비디오 디코딩의 장치로서, 비트스트림으로부터 복수의 이진 심볼들을 디코딩하는 수단 – 상기 복수의 이진 심볼들 중 제1 이진 심볼은 컨텍스트 기반 모드를 이용하여 엔트로피 디코딩되고, 상기 제1 이진 심볼에 후속하는 각각의 이진 심볼은 바이패스 모드에서 엔트로피 디코딩됨 –; 이진화 방식에 대응하는, 상기 복수의 이진 심볼들에 의해 표현된 인덱스를 획득하는 수단; 및 2개의 예측기들의 가중 합으로서 블록에 대한 예측을 형성하는 수단 – 상기 인덱스는 상기 가중 합을 형성할 때 상기 2개의 예측기들을 가중 시에 이용된 개개의 가중 인자들을 표시함 – 를 포함하는 장치가 제공된다.

[0008] 다른 실시예에 따르면, 비디오 인코딩의 장치로서, 인코딩될 블록에 액세스하는 수단; 2개의 예측기들의 가중 합으로서 상기 블록에 대한 예측을 형성하는 수단; 및 상기 가중 합을 형성할 때 상기 2개의 예측기들을 가중 시에 이용된 개개의 가중 인자들을 표시하기 위한 인덱스를 인코딩하는 수단을 포함하고, 상기 인덱스는 이진화 방식을 이용하여 복수의 이진 심볼들로 이진화되고, 상기 복수의 이진 심볼들 중 제1 이진 심볼은 컨텍스트 기반 모드를 이용하여 엔트로피 인코딩되고, 상기 제1 이진 심볼에 후속하는 각각의 이진 심볼은 바이패스 모드에서 엔트로피 인코딩되는 장치가 제공된다.

도면의 간단한 설명

[0009] 도 1은 비디오 인코더의 실시예의 블록도를 도시한다.

도 2는 압축된 HEVC 픽처를 표현하기 위한 코딩 트리 유닛 및 코딩 트리 개념들을 도시하는 도면 예이다.

도 3은 코딩 트리 유닛을 코딩 유닛들, 예측 유닛들 및 변환 유닛들로 분할하는 것을 도시하는 도면 예이다.

도 4는 비디오 디코더의 실시예의 블록도를 도시한다.

도 5는 CABAC 디코딩 프로세스를 도시한다.

도 6은 CABAC 인코딩 프로세스를 도시한다.

도 7은 QTBT(Quad-Tree Plus Binary-Tree) CTU 표현을 도시하는 도면 예이다.

도 8은 코딩 유닛 분할 모드들을 도시하는 도면 예이다.

도 9는 VVC 드래프트 4에서 GBi 인덱스를 코딩하는 프로세스를 도시한다.

도 10은 VVC 드래프트 4에서 GBi 인덱스를 파싱하는 프로세스를 도시한다.

도 11은 실시예에 따른 GBi 인덱스를 코딩하는 프로세스를 도시하고, 도 11a는 다른 실시예에 따른 GBi 인덱스를 코딩하는 프로세스를 도시한다.

도 12는 실시예에 따른 GBi 인덱스를 파싱하는 프로세스를 도시하고, 도 12a는 다른 실시예에 따른 GBi 인덱스를 파싱하는 프로세스를 도시하고, 도 12b 및 도 12c는 각각 비-저지연 모드(non-low-delay mode) 및 저지연 모드(low-delay mode)에서의 gbiCodingIndex 이진화 및 코딩/파싱 프로세스를 도시하고, 도 12d는 디폴트 GBI 모드를 시그널링하는 제1 빈이 0과 동일할 때의 트리들을 도시한다.

도 13은 AMVP 모드에서의 모션 벡터 예측기 후보 리스트 구성을 도시하는 도면 예이다.

도 14는 AMVP 후보 리스트를 구성하기 위해 고려된 공간적 및 시간적 모션 벡터 예측 후보들을 도시하는 도면 예이다.

도 15는 VVC 드래프트 4에서의 AMVP에서의 모션 벡터 예측 인덱스의 CABAC 코딩을 도시한다.

도 16은 실시예에 따른, mvp_l0_flag 및 mvp_l1_flag 신택스 요소들을 바이패스 코딩하는 프로세스를 도시한다.

도 17은 VVC 드래프트 5에서의 SBT 툴의 원리를 도시한다.

도 18은 VVC 드래프트 5에서 INTER 코딩 유닛의 SBT 모드를 디코딩하는 프로세스를 도시한다.

도 19a는 실시예에 따른 SBT 모드 디코딩 프로세스를 도시하고, 도 19b는 다른 실시예에 따른 SBT 모드 디코딩 프로세스를 도시한다.

도 20a는 4x8 및 8x4 INTRA 코딩 유닛들에 대한 허용된 ISP 분할들을 도시하고, 도 20b는 4x8 또는 8x4와 다른 크기를 갖는 INTRA 코딩 유닛들에 대한 허용된 ISP 분할들을 도시한다.

도 21은 다수 참조 라인 인트라 예측의 원리를 도시한다.

도 22는 VVC 드래프트 5에서 'intra_luma_ref_idx' 신택스 요소를 파싱하는 프로세스를 도시한다.

도 23은 실시예에 따른, 'intra_luma_ref_idx' 신택스 요소를 파싱하는 단순화된 프로세스를 도시한다.

도 24는 본 실시예들의 양태들이 구현될 수 있는 시스템의 블록도를 도시한다.

발명을 실시하기 위한 구체적인 내용

[0010] 도 1은 고효율 비디오 코딩(HEVC) 인코더와 같은 예시적인 비디오 인코더(100)를 도시한다. 도 1은 또한 JVET(Joint Video Exploration Team)에 의해 개발 중인 VVC(Versatile Video Coding) 인코더와 같은, HEVC 표준 또는 HEVC와 유사한 기술들을 이용하는 인코더에 대해 개선들이 이루어지는 인코더를 도시할 수 있다.

[0011] 본 출원에서, 용어들 "재구성된" 및 "디코딩된"은 상호교환가능하게 이용될 수 있고, 용어들 "인코딩된" 또는 "코딩된"은 상호교환가능하게 이용될 수 있으며, 용어들 "이미지", "픽처" 및 "프레임"은 상호교환가능하게 이용될 수 있다. 필수적인 것은 아니지만 통상적으로, 용어 "재구성된"은 인코더 측에서 이용되는 한편, "디코딩된"은 디코더 측에서 이용된다.

[0012] 인코딩되기 전에, 비디오 시퀀스는 사전-인코딩 처리(101), 예를 들어, 입력 컬러 픽처에 컬러 변환을 적용하는 것(예를 들어, RGB 4:4:4로부터 YCbCr 4:2:0로의 변환), 또는 (예를 들면, 컬러 성분들 중 하나의 히스토그램 등화(histogram equalization)를 이용하여) 압축에 더 탄력적인 신호 분산을 얻기 위하여 입력 픽처 컴포넌트들의 리맵핑(remapping)을 수행하는 것을 거칠 수 있다. 메타데이터는 전처리에 연관되고, 비트스트림에 첨부될 수 있다.

[0013] 비디오 시퀀스를 하나 이상의 픽처로 인코딩하기 위해, 픽처는, 예를 들어, 하나 이상의 슬라이스로 파티셔닝되고(102), 여기서 각각의 슬라이스는 하나 이상의 슬라이스 세그먼트를 포함할 수 있다. HEVC에서, 슬라이스 세

그먼트는 코딩 유닛들, 예측 유닛들 및 변환 유닛들로 조직된다. HEVC 사양은 "블록들"과 "유닛들" 사이를 구별하며, 여기서 "블록"은 샘플 어레이에서의 특정 영역(예를 들어, 루마(luma), Y)을 처리하고, "유닛"은 모든 인코딩된 컬러 성분들(Y, Cb, Cr, 또는 단색)의 병치된 블록들, 및 블록들과 연관되는 신택스 요소들 및 예측 데이터(예를 들어, 모션 벡터들)를 포함한다.

[0014] HEVC에 따른 코딩의 경우, 도 2에 도시된 바와 같이, 픽처는 구성가능한 크기(전형적으로 64x64, 128x128, 또는 256x256 픽셀)를 갖는 정사각형 형상의 코딩 트리 블록(CTB)들로 파티셔닝되고, 코딩 트리 블록들의 연속적인 세트는 슬라이스로 그룹화된다. 코딩 트리 유닛(CTU)은 인코딩된 컬러 성분들의 CTB들을 포함한다. CTB는 코딩 블록(CB)들로 파티셔닝하는 4진 트리(quadtree)의 루트이고, 코딩 블록은 하나 이상의 예측 블록(PB)으로 파티셔닝될 수 있고, 도 3에 도시된 바와 같이, 변환 블록(TB)들로 파티셔닝하는 4진 트리의 루트를 형성한다. 코딩 블록, 예측 블록 및 변환 블록에 대응하여, 코딩 유닛(CU)은 예측 유닛(PU)들 및 변환 유닛(TU)들의 트리-구조화된 세트를 포함하고, PU는 모든 컬러 성분들에 대한 예측 정보를 포함하고, TU는 각각의 컬러 성분에 대한 잔차 코딩 신택스 구조를 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 이용될 수 있다. 또한, "블록"은 H.264/AVC 또는 다른 비디오 코딩 표준들에 명시된 바와 같이 매크로블록 및 파티션을 지칭하기 위해, 그리고 더 일반적으로는 다양한 크기들의 데이터의 어레이를 지칭하기 위해 또한 이용될 수 있다.

[0015] 인코더(100)에서, 픽처는 아래에서 설명되는 바와 같이 인코더 요소들에 의해 인코딩된다. 인코딩될 픽처는, 예를 들어, CU들의 단위로 처리된다. 각각의 코딩 유닛은 인트라 또는 인터 모드를 이용하여 인코딩된다. 코딩 유닛이 인트라 모드에서 인코딩될 때, 그것은 인트라 예측(160)을 수행한다. 인터 모드에서는, 모션 추정(motion estimation)(175) 및 보상(compensation)(170)이 수행된다. 인코더는 코딩 유닛을 인코딩하기 위해 인트라 모드 또는 인터 모드 중 어느 것을 이용할지를 결정(105)하고, 예측 모드 플래그에 의해 인트라/인터 결정을 표시한다. 예측 잔차들은 원래 이미지 블록으로부터 예측된 블록을 감산(110)함으로써 계산된다.

[0016] 예측 잔차들이 이어서 변환(125) 및 양자화(130)된다. 양자화된 변환 계수들 뿐만 아니라 모션 벡터들(motion vectors) 및 다른 신택스 요소들은, 비트스트림을 출력하도록 엔트로피 코딩(145)된다. 비제한적인 예로서, 컨텍스트 기반 적응적 이진 산술 코딩(context-based adaptive binary arithmetic coding)(CABAC)은 신택스 요소들을 비트스트림으로 인코딩하는데 이용될 수 있다.

[0017] CABAC로 인코딩하기 위해, 비-이진(non-binary) 신택스 요소 값은 이진화 프로세스를 통해, 빈 스트링(bin string)이라고 불리는 이진 시퀀스에 맵핑된다. 빈에 대해, 컨텍스트 모델이 선택된다. "컨텍스트 모델"은 하나 이상의 빈에 대한 확률 모델이고, 최근에 코딩된 심볼의 통계에 따라 이용가능한 모델의 선택으로부터 선택된다. 각각의 빈의 컨텍스트 모델은 컨텍스트 모델 인덱스("컨텍스트 인덱스"라고도 함)에 의해 식별되고, 상이한 컨텍스트 인덱스들은 상이한 컨텍스트 모델들에 대응한다. 컨텍스트 모델은 각각의 빈이 '1' 또는 '0'일 확률을 저장하고, 적응적 또는 정적일 수 있다. 정적 모델은 빈들 '0' 및 '1'에 대해 동일한 확률로 코딩 엔진을 트리거한다. 적응적 코딩 엔진에서, 컨텍스트 모델은 빈의 실제 코딩된 값에 기초하여 업데이트된다. 적응적 및 정적 모델들에 대응하는 동작 모드들은 각각 정규 모드 및 바이패스 모드라고 지칭된다. 컨텍스트에 기초하여, 이진 산술 코딩 엔진은 대응하는 확률 모델에 따라 빈을 인코딩 또는 디코딩한다.

[0018] 인코더는 또한 변환을 스킵하고, 예를 들어, 4x4 TU 기반으로, 비변환된 잔차 신호에 직접 양자화를 적용할 수 있다. 인코더는 또한 변환 및 양자화 둘다를 바이패스할 수 있는데, 즉, 잔차는 변환 또는 양자화 프로세스의 적용 없이 직접 코딩된다. 직접 PCM 코딩에서는, 예측이 적용되지 않으며, 코딩 유닛 샘플들이 비트스트림으로 직접 코딩된다.

[0019] 인코더는 추가적 예측들을 위한 기준을 제공하기 위해 인코딩된 블록을 디코딩한다. 예측 잔차들을 디코딩하기 위해, 양자화된 변환 계수들은 역 양자화(140) 및 역 변환(150)된다. 디코딩된 예측 잔차들과 예측된 블록을 결합(155)하면, 이미지 블록이 재구성된다. 루프 내(in-loop) 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.

[0020] 도 4는 HEVC 디코더와 같은 예시적인 비디오 디코더(200)의 블록도를 도시한다. 디코더(200)에서, 비트스트림은 아래에서 설명되는 바와 같이 디코더 요소들에 의해 디코딩된다. 비디오 디코더(200)는 일반적으로, 비디오 데이터를 인코딩하는 것의 일부로서 비디오 디코딩을 수행하는 도 1에 설명된 바와 같은 인코딩 패스(pass)에 대해 역인 디코딩 패스를 수행한다. 도 4는 또한 HEVC 표준 또는 VVC 디코더와 같은 HEVC와 유사한 기술들을

이용하는 디코더에 대한 개선들이 이루어지는 디코더를 도시할 수 있다.

[0021] 특히, 디코더의 입력은 비디오 인코더(100)에 의해 생성될 수 있는 비디오 비트스트림을 포함한다. 변환 계수들, 모션 벡터들, 픽처 파티셔닝 정보, 및 다른 코딩된 정보를 획득하기 위해 비트스트림은 먼저 엔트로피 디코딩(230)된다. CABAC가 엔트로피 코딩을 위해 이용된다면, 컨텍스트 모델은 인코더 컨텍스트 모델들과 동일한 방식으로 초기화되고, 컨텍스트 모델들에 기초하여 비트스트림으로부터 신택스 요소들이 디코딩된다.

[0022] 도 5는 입력 코딩된 비트스트림이 주어진 신택스 요소의 CABAC 디코딩 프로세스를 도시한다. 이것은 도 6의 신택스 요소 코딩 프로세스의 역 프로세스이다.

[0023] 도 5의 프로세스에 대한 입력은 HEVC 또는 VVC와 같은 비디오 압축 표준에 전형적으로 부합하는 코딩된 비트스트림을 포함한다. 디코딩 프로세스의 임의의 시점에서, 디코더는 어느 신택스 요소가 다음에 디코딩되어야 하는지를 안다. 이것은 표준화된 비트스트림 신택스 및 디코딩 프로세스에서 완전히 명시된다. 또한, 디코딩할 현재 신택스 요소가 어떻게 이진화되는지(즉, 빈들이라고 지칭되는 이진 심볼들의 시퀀스로서 표현되고, 각각은 '1' 또는 '0'과 동일함), 및 빈 스트링의 각각의 빈이 어떻게 인코딩되었는지를 또한 안다.

[0024] 따라서, CABAC 디코딩 프로세스의 제1 스테이지(도 5의 좌측)는 일련의 빈들을 디코딩한다. 각각의 빈에 대해, 디코더는 빈이 바이패스 모드 또는 정규 모드에 따라 인코딩되었는지를 안다. 바이패스 모드는 비트스트림에서의 비트를 단순히 판독하고, 그렇게 획득된 비트 값을 현재 빈에 할당하는 것으로 구성된다. 이 모드는 간단하고 따라서 빠르다는 이점을 갖는다. 따라서, 이것은 균일한 통계적 분포, 즉, '1' 또는 '0'과 동일할 동일한 확률을 갖는 빈들에 대해 전형적으로 효율적으로 사용된다.

[0025] 반대로, 현재 빈이 바이패스 모드에서 코딩되지 않으면, 그것은 빈이 소위 정규 모드에서, 즉, 컨텍스트 기반 산술 코딩을 통해 코딩되었음을 의미한다. 이 경우, 고려된 빈의 디코딩은 다음과 같이 진행된다. 먼저, 컨텍스트 모델러 모듈(context modeler module)을 이용하여 현재 빈의 디코딩을 위한 컨텍스트가 획득된다. 컨텍스트의 목표는, 일부 컨텍스트 또는 이전 정보 X가 주어지면, 현재 빈이 값 '0'을 갖는 조건부 확률을 획득하는 것이다. 여기서 이전 X는, 현재 빈이 디코딩되고 있을 때, 동기 방식으로 인코더 측과 디코더 측 양쪽 모두에서 이용가능한, 일부 이미 디코딩된 신택스 요소의 값일 수 있다.

[0026] 전형적으로, 빈의 디코딩을 위해 이용된 이전 X는 표준에 명시되어 있고, 디코딩할 현재 빈과 통계적으로 상관되어 있기 때문에 선택된다. 이 컨텍스트 정보를 이용하는 것의 관심은, 그것이 빈을 코딩하는 레이트 비용을 감소시킨다는 것이다. 이것은, 빈과 X가 더 상관될 때, X가 주어진 빈의 조건부 엔트로피가 더 낮다는 사실에 기초한다. 이하의 관계는 정보 이론에서 잘 알려져 있다: H (빈|X) < H(빈)

[0027] 이는 X를 아는 빈의 조건부 엔트로피가 빈과 X가 통계적으로 상관되는 경우 빈의 엔트로피보다 낮다는 것을 의미한다. 따라서, 컨텍스트 정보 X는 빈이 '0' 또는 '1'일 확률을 획득하기 위해 이용된다. 이러한 조건부 확률들이 주어지면, 정규 디코딩 엔진은 이진 값 빈의 산술 디코딩을 수행한다. 그 후, 빈의 값은 현재 컨텍스트 정보 X를 아는 현재 빈과 연관된 조건부 확률들의 값을 업데이트하는데 이용된다. 이것은 컨텍스트 모델 업데이트라고 지칭된다. 빈들이 디코딩(또는 코딩)되고 있는 한 각각의 빈에 대한 컨텍스트 모델을 업데이트하는 것은, 각각의 이진 요소에 대한 컨텍스트 모델링을 점진적으로 세분화(refining)하는 것을 허용한다. 따라서, CABAC 디코더는 각각의 정규 인코딩된 빈의 통계적 거동을 점진적으로 학습한다. 컨텍스트 모델러 및 컨텍스트 모델 업데이트 단계들은 인코더 측 및 디코더 측에서 엄격히 동일한 동작들이라는 점에 유의한다.

[0028] 현재 빈의 정규 산술 디코딩 또는 그것의 바이패스 디코딩은, 코딩된 방법에 따라, 일련의 디코딩된 빈들을 초래한다.

[0029] 이어서, 도 5의 우측에 도시된 CABAC 디코딩의 제2 단계는 이 일련의 이진 심볼들을 신택스 요소들로 변환한다. 신택스 요소는 플래그의 형태를 취할 수 있고, 이 경우 그것은 현재 디코딩된 빈들의 값을 직접 취한다. 다른 측에서, 현재 신택스 요소의 이진화가 고려된 표준 사양에 따라 여러 빈들의 세트에 대응하면, 이진 코드워드의 신택스 요소로의 변환이 발생한다.

[0030] 이것은 인코더에 의해 행해진 이진화 단계의 역을 진행한다. 따라서, 여기서 수행된 역 변환은 이들 신택스 요소들의 값을 그들 개개의 디코딩된 이진화된 버전들에 기초하여 획득한다.

[0031] 예를 들어, 마지막 유효 계수 위치(last significant coefficient position) 및 병합 인덱스(merge index)(병합 후보들의 리스트에서의 후보의 위치를 표시함)의 프리픽스 코드에 대응하는 신택스 요소들은 절단 라이스 이진화(truncated Rice binarization)를 이용하여 이진화된다. 마지막 유효 계수 위치 플래그에 대해, 모든 빈들

은 정규 모드에서 인코딩되고; 병합 인덱스에 대해, 제1 빈은 정규 모드에서 인코딩되고, 다른 빈들은 바이패스 모드에서 인코딩된다.

[0032] 픽처 파티셔닝 정보는 픽처가 어떻게 파티셔닝되는지, 예를 들어, CTU들의 크기, 및 CTU가 CU들로, 그리고 적용 가능할 때 가능하게는 PU들로 분할되는 방식을 표시한다. 따라서, 디코더는 디코딩된 픽처 파티셔닝 정보에 따라 픽처를, 예를 들어, CTU들로, 그리고 각각의 CTU를 CU들로 분할(235)할 수 있다. 변환 계수들은 예측 잔차들을 디코딩하기 위해 역 양자화(240) 및 역 변환(250)된다.

[0033] 디코딩된 예측 잔차들과 예측된 블록을 결합(255)하면, 이미지 블록이 재구성된다. 예측된 블록은 인트라 예측(260) 또는 모션 보상된 예측(즉, 인터 예측)(275)으로부터 획득(270)될 수 있다. 루프 내 필터들(265)은 재구성된 이미지에 대해 적용된다. 필터링된 이미지는 참조 픽처 버퍼(280)에 저장된다.

[0034] 디코딩된 픽처는 추가로, 사후-디코딩 처리(285), 예를 들어, 역 컬러 변환(예를 들어, YCbCr 4:2:0으로부터 RGB 4:4:4로의 변환) 또는 사전-인코딩 처리(101)에서 수행된 리맵핑 처리의 역을 수행하는 역 리맵핑을 거칠 수 있다. 사후-디코딩 처리는 사전-인코딩 처리에서 도출되고 비트스트림에서 시그널링되는 메타데이터를 이용할 수 있다.

[0035] 압축 영역에서 보다 유연한 방식으로 픽처 데이터를 표현하기 위해, 압축 영역에서 새로운 코딩 트리 유닛 표현을 포함하는 최근에 새롭게 생겨난 비디오 압축 툴들이 제안된다. 코딩 트리의 더 유연한 표현으로, HEVC 표준의 CU/PU/TU 배열에 비해 압축 효율이 증가될 수 있다.

[0036] 일 예에서, QTBT(Quad-Tree plus Binary-Tree) 코딩 툴은 증가된 유연성을 제공하는 새로운 툴이다. QTBT 코딩 트리에서, 코딩 유닛들은 4진 트리 및 이진 트리 방식 둘다로 분할될 수 있다. QTBT 기술에서, CU는 정사각형 또는 직사각형 형상을 갖는다. 코딩 유닛의 크기는 항상 2의 거듭 제곱이고, 전형적으로 4 내지 128이다. 코딩 트리 유닛의 QTBT 코딩 트리 표현의 예가 도 7에 도시된다.

[0037] 코딩 유닛의 분할은 최소 레이트 왜곡 비용으로 CTU의 QTBT 표현을 결정함으로써 수행될 수 있는 레이트 왜곡 최적화 절차를 통해 인코더 측에서 결정된다. CTU의 QTBT 분해는 2개의 스테이지로 이루어진다: 먼저 CTU는 4진 트리 방식으로 분할되고, 그 후 각각의 4진 트리 리프는, 실선들이 4진 트리 분해 단계를 나타내고 파선들이 4진 트리 리프들에 공간적으로 내장된 이진 분해를 나타내는 도 7에 도시된 바와 같이, 이진 방식으로 추가로 분할될 수 있다. 인트라 슬라이스들에서, 루마 및 크로마 블록 파티셔닝 구조는 분리되고, 독립적으로 결정된다.

[0038] 예측 유닛들 또는 변환 유닛으로의 CU 파티셔닝은 더 이상 이용되지 않는다. 즉, 각각의 코딩 유닛은 단일 예측 유닛(2Nx2N 예측 유닛 파티션 타입)과 단일 변환 유닛(변환 트리로의 분할 없음)으로 체계적으로 만들어진다.

[0039] 수평 또는 수직 트리플 트리 분할 모드(HOR_TRIPLE, VER_TRIPLE)로 지칭되는, VVC(Versatile Video Coding) 비디오 압축 표준에서 채택된 추가적인 CU 분할 모드는, 도 8에 도시된 바와 같이, 코딩 유닛(CU)을 고려된 공간 분할의 방향으로 부모 CU 크기의 1/4, 1/2 및 1/4과 동일한 개개의 크기들을 갖는 3개의 서브 코딩 유닛(sub-CU)들로 분할하는 것으로 구성된다.

[0040] 본 실시예들은 신택스 요소들의 인코딩 및 디코딩에 관한 것이다. 일부 실시예들에서, 일부 인트라 또는 인터 예측 파라미터들의 엔트로피 코딩은 복잡도를 감소시키기 위해 수정된다.

[0041] 전술한 바와 같이, 많은 이진 심볼들(또는 빈들)이 컨텍스트 적응적 이진 산술 코딩 프로세스를 통해 코딩 및 디코딩된다. 이 프로세스는 빈들을 코딩하는 2가지 방식: 정규 모드 및 바이패스 모드를 수반한다. 바이패스 코딩 모드는 정규 코딩 모드보다 훨씬 덜 복잡하다. 따라서, 바이패스 코딩에 비해 정규 코딩 모드에서 코딩 효율 개선이 전혀 또는 거의 얻어지지 않을 때, 정규 코딩 프로세스를 바이패스 코딩 프로세스로 대체하는 것이 관심사이다.

[0042] 일 실시예에서, 바이패스 코딩 프로세스는 일반화된 양방향 예측 인덱스를 나타내는 빈 스트링의 제1의 하나 이상의 빈을 시그널링하는데 이용된다. 다른 실시예에서, 바이패스 코딩 프로세스는, 정규 코딩 프로세스 대신에, AMVP 모드에서 코딩된 코딩 유닛(CU)의 모션 벡터를 인코딩하기 위해 어느 모션 벡터 예측기가 이용되는지를 시그널링하기 위해 이용된다.

[0043] 실제로, 현재 VVC 코딩 시스템에서 이들 빈에 대한 컨텍스트 기반 산술 코딩의 이용에도 불구하고 이들 신택스 요소에서 인코딩할 빈의 조건부 엔트로피는 1 비트의 정보에 가깝다는 것이 검출되었다. 또한, 실험 결과들은

제안된 바이패스 코딩을 이용함으로써 VVC 압축 성능에 미치는 영향은 무시할만한 하다는 것을 보여준다.

[0044] 이하에서, 일반화된 양방향 예측 인덱스 및 모션 벡터 예측기의 시그널링을 더 상세히 설명한다.

[0045] 일반화된 양방향 예측 인덱스의 시그널링

[0046] **VVC 드래프트 4에서의 일반화된 양방향 예측**

[0047] VVC 드래프트 4("Versatile Video Coding(Draft 4)", B. Bross et al., 13th JVET meeting, January 9-18, 2019, Marrakech 참조)에서, 인터 CU는 소위 일반화된 양방향 예측(GBi)의 이용을 통해 시간적으로 예측될 수 있다. 일반화된 양방향 예측에서, 양방향 예측된 블록의 시간적 예측은 다음의 표현식에 따라 2개의 참조 블록들의 가중 평균으로서 계산된다:

[0048] $$P_{bipred} = \left((8 - w) \times P_0 + w \times P_1 + 4\right) \gg 3$$

[0049] 여기서, P_0 및 P_1은 시간적 예측기들이고, w는 다음의 세트에서 선택된다:

[0050] - (과거의 모든 그들의 참조 픽처들을 갖는) 저지연 픽처들에 대해: w∈{-2, 3, 4, 5, 10}. (적어도 하나의 과거 및 하나의 미래 참조 픽처를 갖는) 비-저지연 픽처들에 대해: w∈{3, 4, 5}

[0051] - GBi 가중치 w는 인코더 측에서 레이트 왜곡 최적화 프로세스에 기초하여 선택되고, 비트스트림에서 시그널링된다. GBi는 또한 아핀(affine) 모션 보상 또는 적응적 모션 벡터 해상도와 같은, VVC에서의 다양한 모션 보상 툴들과 결합된다.

[0052] CU를 예측하는데 이용된 GBi 가중치를 시그널링하는 신택스 요소는 GBi 인덱스라고 지칭된다.

[0053] **VVC 드래프트 4에서의 GBi 인덱스의 코딩(인코더)**

[0054] VVC 드래프트 4에서, GBi 인덱스는 먼저 다른 인덱스 gbiCodingIndex로 변환된다. 디폴트 가중치(동일한 가중치)가 이용될 것인지를 나타내기 위해 플래그가 결정된다. 플래그는 디폴트 가중치 w = 4(예측기들 P_0 및 P_1 둘 다에 대해 동일한 가중치들)에 대해 1로 설정되고, 다른 가중치들에 대해 0으로 설정된다. 나머지 gbiCodingIndex는 절단 라이스(절단 단항(truncated unary)) 빈 스트링을 이용하여 이진화된다. 절단 라이스 빈 스트링의 플래그 또는 각각의 빈은 전용 CABAC 컨텍스트로 CABAC 인코딩된다. 특히, 플래그는 컨텍스트 모델 ID 0으로 (즉, 정규 모드를 이용하여) 컨텍스트 기반 코딩된다. 빈들인 빈1, 빈2 및 빈3은 각각 컨텍스트 모델 ID들 4, 5 및 6으로 컨텍스트 기반 코딩된다.

[0055] 표 1은 저지연 모드에 대한 GBi 인덱스(GBiIdx) 코딩을 도시하며, 여기서 GBi 모드들의 수는 numGBiModes = 5로서 설정되고, 제1 빈에 더하여 빈들의 최대 수는 numBins = 3으로서 설정된다.

[0056] 표 1. 저지연 모드

[0057]

w	GBi 인덱스	gbiCodingIndex	플래그	컨텍스트	빈 스트링	컨텍스트들	gbiParsingOrder
-2	0	4	0	[0]	000	[4, 5, 6]	0
3	1	2	0	[0]	01	[4]	1
4	2	0	1	[0]	---	---	2
5	3	1	0	[0]	1	[4, 5]	3
10	4	3	0	[0]	001	[4, 5, 6]	4

[0058] 표 2는 비-저지연 모드에 대한 GBi 인덱스 코딩을 도시하며, 여기서 GBi 모드의 수는 numGBiModes = 3으로서 설정되고, 제1 빈에 더하여 빈들의 최대 수는 numBins = 1로서 설정된다.

[0059] 표 2. 비-저지연 모드

[0060]

w	GBi 인덱스	gbiCodingIndex	플래그	컨텍스트	빈 스트링	컨텍스트들	gbiParsingOrder
3	1	2	0	[0]	0	[4]	1
4	2	0	1	[0]	---	---	2
5	3	1	0	[0]	1	[4]	3

[0061] 표 1 및 표 2에 도시된 바와 같은 플래그와 빈 스트링의 연결은 gbiCodingIndex의 직접 절단 라이스 이진화로서 간주될 수 있다는 점에 유의해야 한다. 즉, gbiCodingIndex는 절단 라이스(절단 단항) 빈 스트링을 이용하여 이진화된다. 각각의 빈은 전용 CABAC 컨텍스트로 CABAC 인코딩된다. 특히, 제1 빈인 빈0은 컨텍스트 모델 ID 0으로 (즉, 정규 모드를 이용하여) 컨텍스트 기반 코딩된다. 제1 빈은 디폴트 가중치 w = 4(예측기들 P_0 및 P_1 둘다에 대해 동일한 가중치)에 대해 1로 설정되고, 다른 가중치에 대해 0으로 설정된다는 점에 유의한다. 따라서, 제1 빈은 또한 디폴트 가중치가 이용될 것인지를 나타내는 플래그로서 간주될 수 있다. 그 다음, 후속 빈들인 빈1, 빈2 및 빈3은 각각 컨텍스트 모델 ID들 4, 5 및 6으로 컨텍스트 기반 코딩된다.

[0062] 도 9는 VVC 드래프트 4에서와 같은 GBi 인덱스의 코딩을 도시한다. 단계 910에서, 변수 Idx는 0으로 설정된다. 단계 920에서, 현재 CU와 연관된 GBiIdx 값은 표 1 및 표 2의 열 "gbiCodingIndex"에 의해 정의된 표 GbiCodingOrder를 통해 gbiCodingIndex로 변환된다. 단계 930에서, 플래그(gbiCodingIndex==0)가 인코딩되고, 이는 값 gbiCodingIndex가 0과 동일한지를 나타낸다. 0 값은 현재 CU의 GBiIdx 값이 디폴트 양방향 예측 모드에 대응하는 GBI_DEFAULT와 동일한 경우, 즉 w = 4인 경우에 대응한다. gbiCodingIndex의 값이 0이면(940), 프로세스는 종료된다.

[0063] 그렇지 않은 경우, 인코더는 슬라이스가 저지연 타입인지를 체크(945)한다. 슬라이스가 저지연 타입이면, numGbiModes는 5로 설정(950)되고; 슬라이스가 저지연 타입이 아니면, numGbiModes는 3으로 설정(955)된다. 단계 960에서, Idx는 1로 설정된다. 단계 965에서, 컨텍스트 모델 ID contextId는 4로 설정된다. 단계 970에서, numBins는 numGbiModes - 2로 설정된다. gbiCodingIndex는 빈 스트링으로 이진화된다. 그 다음, gbiCodingIndex를 나타내는 빈들은, 모든 빈들이 인코딩될 때까지(990), 대응하는 contextId(985)로 하나씩 인코딩(980)된다. 프로세스는 단계(999)에서 종료한다.

[0064] 도 10은 VVC 드래프트 4에서와 같은 GBi 인덱스의 파싱을 도시한다. 디코더는 비트스트림으로부터 값 "Idx"를 디코딩하고, 디코딩된 값 Idx와 CU-레벨 GBiIdx 파라미터를 링크하는 맵핑 표 GbiParsingOrder를 이용하여, "Idx"를 입력 CU와 연관된 실제 GBiIdx로 변환한다.

[0065] 더 구체적으로, 단계 1010에서, 변수 Idx가 0으로 설정된다. 단계 1020에서, 값 gbiCodingIndex가 0과 동일한지(gbiCodingIdx== 0)를 표시하는 플래그가 contextId=0을 이용하여 디코딩된다. 0 값은 현재 CU의 GBiIdx 값이 디폴트 양방향 예측 모드에 대응하는 GBI_DEFAULT와 동일한 경우, 즉, w= 4인 경우에 대응한다. 디코딩된 심볼이 1인 경우, 디코더는 단계 1090으로 진행한다.

[0066] 그렇지 않은 경우, 플래그가 1이면(1025), 단계 1030에서, Idx는 1로 설정된다. 단계 1035에서, 컨텍스트 모델 ID contextId는 4로 설정된다. 디코더는 슬라이스가 저지연 타입인지를 체크(1040)한다. 슬라이스가 저지연 타입이면, numGbiModes는 5로 설정(1045)되고; 슬라이스가 저지연 타입이 아니면, numGbiModes는 3으로 설정(1050)된다. 단계 1060에서, numBins는 numGbiModes - 2로 설정된다. gbiCodingIndex를 나타내는 빈들은 그 후, 1과 동일한 빈 심볼이 발견되거나 numBins 빈들이 파싱될 때까지(1085), 대응하는 contextId(1080)로 하나씩 반복적으로 디코딩(1070)된다. 각각의 반복에서, 변수 Idx가 증분된다. 단계 1090에서, "Idx"는 표 GbiParsingOrder를 통해 GBi 인덱스로 변환된다. 프로세스는 단계 1099에서 종료한다.

[0067] **제안된 GBi 인덱스 코딩 프로세스(인코더)**

[0068] 표 3 및 표 4는 각각 저지연 모드 및 비-저지연 모드에 대한 GBi 인덱스(GBiIdx) 코딩에 대한 제안된 수정들을 도시한다. 이 제안된 방법에서, 제1 빈은 바이패스 모드(표들에서 "b"로 표현됨)에서 인코딩되고, 다른 빈들은 정규 모드에서 인코딩된다.

[0069] 표 3. 저지연 모드

w	GBi 인덱스	gbiCodingIndex	플래그	컨텍스트	빈 스트링	컨텍스트들	gbiParsingOrder
-2	0	4	0	[0]	000	[b, 5, 6]	0
3	1	2	0	[0]	01	[b, 5]	1
4	2	0	1	[0]	---	---	2
5	3	1	0	[0]	1	[b, 5]	3
10	4	3	0	[0]	001	[b, 5, 6]	4

[0070]

[0071] 표 4. 비-저지연 모드

w	GBi 인덱스	gbiCodingIndex	플래그	컨텍스트	빈 스트링	컨텍스트들	gbiParsingOrder
3	1	2	0	[0]	0	[b]	1
4	2	0	1	[0]	---	---	2
5	3	1	0	[0]	1	[b]	3

[0072]

[0073] 도 11은 실시예에 따른, 제안된 GBiIdx 파라미터 코딩 프로세스를 도시한다. 단계 1165에서, contextId는 5로 설정된다. 단계 1170에서, 표 3 또는 표 4의 절단 라이스 빈 스트링의 제1 빈은 컨텍스트 기반 산술 코딩 모드 대신에 바이패스 모드에서 인코딩된다. 다른 빈들은, 만약 있다면, 도 9에서 설명된 것들과 유사하게, 정규 모드에서 인코딩된다. 이 제안된 실시예의 이점은 CU-레벨 GBiIdx 파라미터를 인코딩함에 있어서 감소된 복잡도이다.

[0074] **제안된 GBi 파싱 프로세스(디코더)**

[0075] 도 12는 실시예에 따른, GBiIdx 파라미터 파싱 프로세스에 대한 제안된 수정을 도시한다. 알 수 있는 바와 같이, 제안된 GBiIdx 파싱 프로세스의 제1 단계들은 도 10에서 설명된 것들과 동일하다. 다음, 제1 디코딩된 심볼이 0과 동일한 경우(비 GBI_DEFAULT 베이스), 절단 라이스 이진화된 스트링을 나타내는 일련의 빈들이 파싱된다. 제안된 실시예에 따르면, 이 스트링의 제1 빈은 컨텍스트 기반 산술 디코딩 모드 대신에 바이패스 모드를 이용하여 파싱된다. 다른 빈들은 종래 기술에서와 동일한 방식으로 디코딩된다. 이러한 제안된 실시예의 이점은 GBiIdx CU-레벨 파라미터를 파싱함에 있어서 감소된 복잡도이다.

[0076] 상기에서, 인덱스가 다수의 빈들에 의해 표현될 때, 제1 빈은 바이패스 모드에서 코딩되고, 나머지 빈들은 정규 모드에서 인코딩된다. 더 일반적으로, (빈 스트링의 시작에서) 하나보다 많은 빈들이 바이패스 모드에서 인코딩될 수 있고, 나머지 빈들은 정규 모드들에서 인코딩된다. 또한, 상기의 예들에서는 절단 라이스 이진화가 설명되었고, 제안된 코딩 방법들은 다른 이진화 방식들에 적용될 수 있다는 점에 유의해야 한다. 또한, 상기의 신택스 요소의 예로서 GBi 인덱스가 이용된다. 그러나, 전술한 바와 같은 방법들은 다른 신택스 요소들에 적용될 수 있다.

[0077] 일부 다른 신택스 요소들을 인코딩할 때, 예를 들어, 전술한 바와 같이 병합 인덱스를 인코딩할 때, 바이패스 모드 및 정규 모드 둘다가 이용된다는 점에 유의한다. 특히, 병합 인덱스를 인코딩하기 위해, 제1 빈은 정규 모드에서 인코딩되는데, 그 이유는 아마도 제1 빈을 컨텍스트 정보로 인코딩하는 것이 더 효율적이기 때문이다. 나머지 빈들에 대해, 확률은 더 랜덤하게 분포될 수 있고, 동일한 확률에 대응하는 바이패스 모드도 마찬가지로 작용할 수 있다. 그러한 인코딩은, 제1 빈이 바이패스 모드에서 인코딩되고 나머지 빈들은 정규 모드에서 인코딩되는 본 실시예에서 제안된 것과는 반대라는 점에 유의해야 한다.

[0078] 다른 실시예에 따르면, 가중 인자 w = 4에 대응하는 디폴트 GBi 인덱스(GBI_DEFAULT)인지를 나타내는 제1 플래그만이 정규 모드에서 인코딩된다. 이 실시예에서, 그 플래그를 뒤따르는 절단 라이스 빈 스트링의 모든 빈들은 바이패스 모드에서 인코딩된다. 이 실시예의 이점은, 잠재적으로 약간 감소된 코딩 효율의 대가로, GBi 인덱스의 코딩 및 파싱에 대한 추가로 감소된 복잡도이다. 도 11a는 본 실시예에 따른 제안된 gbiCodingIndex 코딩 프로세스를 도시한다. 단계 1180에서, 표 3 또는 표 4의 절단 라이스 빈 스트링의 빈들은 컨텍스트 기반 산술 코딩 모드 대신에 바이패스 모드에서 연속적으로 인코딩된다. 이러한 제안된 실시예의 이점은, 비디오 압축 효율의 관점에서 어떠한 페널티도 없이, CU-레벨 GBiIdx 파라미터를 인코딩함에 있어서 더 감소된 복잡도이다.

[0079] 도 12a는 본 실시예에 따른 gbiCodingIndex 파싱 프로세스에 대한 제안된 수정을 도시한다. 알 수 있는 바와 같이, 제안된 gbiCodingIndex 파싱 프로세스의 제1 단계들은 도 10에서 설명된 것들과 동일하다. 다음, 제1 디코딩된 심볼이 0과 동일한 경우(비 GBI_DEFAULT 경우), 절단 라이스 이진화된 스트링을 나타내는 일련의 빈들이 파싱된다. 제안된 실시예에 따르면, 이 스트링의 빈들은 컨텍스트 기반 산술 디코딩 모드 대신에 바이패스 모드(1280)를 이용하여 파싱된다.

[0080] 표 4A는 도 10의 gbiCodingIndex 파싱 방법과 결합된, 도 9의 gbiCodingIndex 코딩 방법을 이용한 VTM-4.0의 성능과 비교하여, gbiCodingIndex 코딩을 위해 (도 11a, 도 12a에 설명된 바와 같은) 이 제안된 실시예를 이용할 때의 VTM-4.0의 압축 성능을 도시한다. 표에 제공된 수치는 동일한 객관적인 비디오 품질에서 제안된 방법의 평균 비트레이트 감소에 대응한다. 따라서, 음수는 비트레이트 감소, 따라서 더 나은 코딩 효율을 나타내는 반면, 양수는 비트레이트 증가, 따라서 감소된 코딩 효율을 나타낸다. Y, U 및 V 열들은 각각 루마, 크로마 Cb

및 크로마 Cr 성분들에 대응한다. 이 실시예는 도 9의 방법에 비해 루마에서의 평균 비트레이트 수정을 초래하지 않으므로, 더 많은 바이패스 코딩된 빈들의 이용으로 인한 감소된 복잡도에도 불구하고, 비디오 압축 효율에 어떠한 페널티도 도입하지 않는다는 것을 알 수 있다.

[0081] 표 4A. GBi 인덱스 코딩 및 파싱을 위한 제안된 실시예(도 11a, 도 12a)의 압축 성능

랜덤 액세스 메인 10			
	Y	U	V
클래스 A1	-0.01%	-0.03%	0.01%
클래스 A2	-0.01%	-0.02%	0.06%
클래스 B	0.01%	-0.02%	0.02%
클래스 C	0.00%	-0.05%	0.00%
클래스 E			
전체	0.00%	-0.03%	0.02%
클래스 D	0.01%	-0.04%	-0.06%
클래스 F (선택적)	0.00%	-0.02%	-0.02%

저지연 B 메인10			
	Y	U	V
클래스 A1			
클래스 A2			
클래스 B	0.01%	-0.14%	0.05%
클래스 C	-0.03%	0.23%	0.05%
클래스 E	0.01%	0.30%	-0.33%
전체	0.00%	0.09%	-0.05%
클래스 D	0.00%	0.11%	-0.31%
클래스 F (선택적)	-0.03%	0.07%	0.14%

[0082]

[0083] 이하에서, 제안된 gbiCodingIndex 코딩 및 파싱 방법이 코딩 효율의 손실을 갖지 않는 이유, 즉, TR(Truncated Rice) 빈 스트링으로부터 발행된 빈들의 바이패스 코딩이 이들 빈들의 산술 코딩만큼 효율적인 것으로 되는 이유가 설명된다.

[0084] 산술 코딩은 샤논(Shannon) 한계, 즉, 그것이 인코딩하는 심볼들의 엔트로피에 매우 가까운 비트레이트를 달성하고, 따라서 최적의 또는 거의 최적의 엔트로피 코딩 방법이라는 것이 알려져 있다.

[0085] 프로세스의 바이패스 코딩 부분의 최적성은 gbiCodingIndex를 코딩하는데 이용되는 이진화가 허프만(Huffman) 트리에 가깝게 대응한다는 것을 암시한다. 이것은 바이패스 모드에서 gbiCodingIndex의 빈들을 코딩하기 위한 TR 이진화가 허프만 트리에 대응한다는 것을 의미한다. 허프만 코딩은 최적의 가변 길이 코딩 방법이다. 더욱이, 허프만 코딩은 특정 조건들이 충족될 때, 시그널링된 빈들의 엔트로피와 동일한 평균 코드 길이를 생성하는 것으로 알려져 있다. 특히, 허프만 코딩은 허프만 트리의 각각의 분기에 연관된 확률들이 다이애딕(dyadic)인 경우, 즉, 2의 음의 거듭 제곱과 동일한 경우 최적이다: $1/2^n$, 여기서 n은 양의 정수 값이다.

[0086] 도 12b 및 도 12c의 2개의 트리들은 각각 비-저지연 모드 및 저지연 인터 픽처 코딩 구성들에서, 제안된 솔루션으로부터 생기는 gbiCodingIndex 이진화 및 코딩/파싱 프로세스를 도시한다. 채워진 노드들은 정규(컨텍스트 기반) 코딩된 빈들에 대응하고, 채워지지 않은 노드들은 바이패스 코딩된 빈들에 대응한다. 컨텍스트 기반 코딩된 빈이 0과 동일할 확률을 x라고 하자. 도 12b 및 도 12c의 트리들의 각각의 에지에 연관된 값들은 이진화된 gbiCodingIndex의 빈들이 1 또는 0과 동일할 확률들에 대응한다. 예를 들어, 이진화된 gbiCodingIndex의 제2 빈이 0과 동일할 확률은 $x \times \frac{1}{2}$ 이고, 이 빈이 1과 동일할 확률도 $x \times \frac{1}{2}$이다. 도 12d는 디폴트 GBI 모드를 시그널링하는 제1 빈이 0과 동일하다는 것을 아는, 이진화된 gbiCodingIndex의 바이패스 코딩된 빈들의 코딩/파싱에 대응하는 트리를 도시한다. 이미 설명된 바와 같이, 이들 빈들의 바이패스 코딩의 최적성은 도 12d의 트리가 최적인 허프만 코딩 트리이고, 따라서 다이애딕 연관 확률 값들을 갖는다.

[0087] 신택스 요소의 절단 라이스(TR) 이진화로부터 생기는 빈 스트링의 확률들이 항상 다이애딕인 것은 아니다. 예를 들어, VVC 사양에는, 절단 라이스 이진화에 후속하여 이진화되고, 신택스 요소들 last_sig_coeff_x_prefix 및 last_sig_coeff_y_prefix와 같은 모든 빈들이 컨텍스트 기반 코딩되는 일부 다른 신택스 요소들이 존재한다. 이들 2개의 파라미터들은 절단 라이스 이진화되고 모든 빈들은 컨텍스트 기반 코딩된다. 바이패스 코딩의 일부는 압축 효율의 손실로 이어질 것이고, 이는 대응하는 바이패스 코딩이 바이패스 코딩된 빈들의 다이애딕 확률 분포를 갖는 최적의 허프만 코딩 프로세스에 대응하지 않을 것임을 나타낸다.

[0088] 이 실시예에서, TR 이진화 후의 gbiCodingIndex의 빈 스트링의 확률들은 다이애딕에 가깝고, 따라서 바이패스 코딩 패스를 적용하는 것은 최적의 허프만 코딩 프로세스에 대응하면서, 정규 모드에서의 산술 코딩보다 더 낮은 계산 복잡도를 초래한다는 것을 인식한다.

[0089] 모션 벡터 예측기의 시그널링

[0090] **VVC 드래프트 4에서의 AMVP 모드에서의 모션 벡터 예측 코딩**

[0091] AMVP 모션 벡터 코딩 모드는 다음의 요소들을 이용하여 CU의 모션 벡터를 코딩하는 것에 관련된다:

[0092] - 양방향 예측 또는 단방향 예측이 현재 CU를 예측하는데 이용되는지, 및 어느 참조 픽처 리스트가 단방향 예측의 경우에 이용되는지를 표시하는 인터 예측 방향.

[0093] - 각각의 관여된 참조 픽처 리스트에서, 현재 CU를 예측하기 위해 어느 참조 픽처(들)가 이용되는지를 표시하는 참조 픽처 인덱스(들).

[0094] - 현재 CU를 예측하는데 이용된 각각의 참조 픽처에 대한 현재 CU의 실제 모션 벡터를 예측하는데 이용된 모션 벡터 예측기. 이 MV 예측기(또는 MVP, 또는 AMVP 후보)는 2개의 후보들을 포함하는 MV 예측기 리스트에서 인코더에 의해 선택된다. 어느 MV 후보가 선택되는지는, 각각의 관련된 참조 픽처 리스트 L0 및 L1에 대해, mvp_l0_flag 및 mvp_l1_flag로 각각 표시된 플래그를 통해 시그널링된다.

[0095] - 각각의 참조 픽처 리스트 L0 및 L1에 대해, 현재 CU의 실제 모션 벡터들과 그들 개개의 모션 벡터 예측기들 사이의 모션 벡터 차이들.

[0096] AMVP 모션 벡터 예측 후보 리스트의 구성은 도 13 및 도 14에 도시된다. 이 프로세스는 기본적으로 현재 CU 주위의 5개의 공간 위치들 중에서 최대 2개의 공간 후보들을 선택하고, 그 후 이들을 프루닝(pruning)하고, 이들 중 최대 2개를 유지하는 것으로 구성된다. 다음, 현재 슬라이스의 소위 병치된 슬라이스에서, 하부 우측 위치 H에 대응하는 공간 위치들에서, 또는 이용가능하지 않다면, 중심 위치 '중심'에서, 시간 MV 예측 후보가 구해진다. 다음, 프루닝 프로세스가 공간적 후보와 시간적 후보 사이에 적용되고, 리스트는 전체 2개의 요소까지 0 모션 벡터로 채워진다. 결국, AMVP 후보 리스트는 정확히 2개의 모션 벡터 예측 후보들을 포함한다.

[0097] 따라서, 각각 mvp_l0_flag 및 mvp_l1_flag로 표시된 단일 플래그가 비트스트림에서 시그널링되고, 디코더 측에서 파싱되어, AMVP 리스트에 포함된 2개의 요소들 중에서, 어느 AMVP 후보가 각각의 참조 픽처 리스트 L0 및 L1에서 현재 CU의 모션 벡터를 예측하는데 이용되는지를 표시한다.

[0098] 고려된 참조 픽처 리스트 L0 또는 L1에 따라, 플래그 mvp_l0_flag 또는 mvp_l1_flag를 파싱하는 프로세스가 도 15에 도시되어 있다. 이는 이진 심볼 '심볼'의 컨텍스트 기반 산술 디코딩(1510)으로 구성된다. 이것은 단일 CABAC 컨텍스트를 이용한다. 고려된 참조 픽처 리스트 내의 현재 PU 또는 CU의 MV 예측기 인덱스는 디코딩된 심볼의 값을 부여받는다(1520)(PU는 HEVC에서 이용되는 바와 같이 현재 CU 내부의 모션 파티션일 수 있고; VVC 드래프트 4에서는 PU 파티셔닝이 이용되지 않으므로, PU는 CU에 대응한다).

[0099] **제안된 방법: 바이패스 모드에서 mvp_l0_flag 및 mvp_l1_flag를 코딩**

[0100] 다수의 코딩된 비디오 시퀀스들에 걸쳐, mvp_l0_flag 및 mvp_l1_flag 신택스 요소를 CABAC 시그널링하는데 이용된 평균 엔트로피는 1 비트의 정보에 매우 가깝고, 이는, 간단한 바이패스 코딩 프로세스와 비교하여, 이들 플래그의 CABAC 코딩에 의해 어떠한 이점도 야기되지 않는다는 것을 의미한다.

[0101] 따라서, 본 실시예에서, 바이패스 코딩 모드를 통해 이러한 플래그들을 인코딩 및 파싱하는 것이 제안된다. 도 16은 하나의 주어진 참조 픽처 리스트에 대해, 하나의 CU 또는 PU의 MV 예측기 인덱스를 디코딩(1610, 1620)하는 제안된 파싱 프로세스를 도시한다. 도 15의 프로세스와의 차이는 관련된 빈의 CABAC 디코딩이 이 빈의 바이패스 디코딩(1610)으로 대체된다는 것이다. 인코더 측에서, mvp_l0_flag 또는 mvp_l1_flag를 나타내는 이진 심볼은 바이패스 모드에서 인코딩된다. 이 실시예의 이점은 VVC의 코딩 및 파싱 프로세스에서의 복잡도가 감소되고, 코딩 효율에 미치는 영향은 무시할만한 하다.

[0102] 추가 실시예에 따르면, mvp_l0_flag는 고려된 CU가 SMVD(Symmetrical Motion Vector Difference) 모드에서 코딩되는지 여부에 따라 상이한 방식으로 코딩될 수 있다. VVC의 SMVD 모션 벡터 코딩 프로세스는 제1 참조 픽처 리스트 L0에 관한 주어진 CU의 모션 벡터 차이를 코딩하는 것으로 구성된다. 이어서, 다른 참조 픽처 리스트(L1) 내의 고려된 CU의 모션 벡터 차이가 L0 모션 벡터 차이로부터 추론된다. 실제로, 이 모드에서, 2개의 모션 벡터는 대칭이다. L1 모션 벡터는 두 성분들 모두에서 L0 모션 벡터의 반대와 같다. 이 실시예에서, SMVD 경우의 MVP 후보는 정규 모드에서 인코딩될 수 있는 반면, 전통적인 AMVP 모션 벡터 코딩 모드에서의 MVP 후보는 바이패스 모드에서 코딩될 수 있다. 이것은 2개의 별개의 신택스 요소들, 고전적인 mvp_l0_flag 및 smvd_mvp_l0 플래그가 특정될 수 있다는 것을 의미하며, 후자는 SMVD 모드에서 CU의 모션 벡터를 예측하는데 이용된 MV를 특정하는데 이용된다.

[0103] 다른 실시예에 따르면, 고전적인 mvp_l0_flag는 정규 모드에서 코딩될 수 있는 반면, smvd_mvp_l0 플래그는 바이패스 모드에서 코딩될 수 있다. 변형에 따르면, 고전적인 mvp_l0_flag는 정규 모드에서 코딩될 수 있고,

smvd_mvp_l0 플래그 또한 정규 모드에서 코딩될 수 있지만, 통상적인 mvp_l0_flag를 코딩하기 위해 이용된 CABAC 컨텍스트와는 별개의 컨텍스트를 갖는다.

[0104] VVC 드래프트 4 신택스를 예들로서 이용하여, 표 5는 위에서 제안된 방법에 의해 신택스에 대해 초래되는 수정들의 예를 열거한다. 특히, 신택스 요소들 mvp_l0_flag 및 mvp_l1_flag에 대한 서술자는 ae(v)로부터 u(1)로 변경되고, 여기서 ae(v)는 컨텍스트 적응적 산술 엔트로피 코딩된 신택스 요소를 표시하고, u(n)은 n 비트를 이용하는 무부호 정수를 표시한다.

[0105] 표 5

[0106]

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	서술자
…	
} else if(treeType != DUAL_TREE_CHROMA) { /* MODE_INTER or MODE_IBC */	
if(cu_skip_flag[x0][y0] = = 0)	
merge_flag[x0][y0]	ae(v)
if(merge_flag[x0][y0]) {	
merge_data(x0, y0, cbWidth, cbHeight)	
} else if (CuPredMode[x0][y0] = = MODE_IBC) {	
mvd_coding(x0, y0, 0, 0)	
~~**mvp_l0_flag**[x0][y0]~~	~~ae(v)~~
mvp_l0_flag[x0][y0]	u(1)
if(sps_amvr_enabled_flag && (MvdL0[x0][y0][0] != 0 || MvdL0[x0][y0][1] != 0)) {	
amvr_precision_flag[x0][y0]	ae(v)

}	
} else {	
if(tile_group_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_BI && !inter_affine_flag[x0][y0] && RefIdxSymL0 > −1 && RefIdxSymL1 > −1)	
sym_mvd_flag[x0][y0]	ae(v)
if(inter_pred_idc[x0][y0] != PRED_L1) {	
if(NumRefIdxActive[0] > 1 && !sym_mvd_flag[x0][y0])	
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)~~
mvp_l0_flag[x0][y0]	u(1)
} else {	
MvdL0[x0][y0][0] = 0	
MvdL0[x0][y0][1] = 0	
}	
if(inter_pred_idc[x0][y0] != PRED_L0) {	
if(NumRefIdxActive[1] > 1 && !sym_mvd_flag[x0][y0])	
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 {	
if(sym_mvd_flag[x0][y0]) {	
MvdL1[x0][y0][0] = −MvdL0[x0][y0][0]	
MvdL1[x0][y0][1] = −MvdL0[x0][y0][1]	
} else	

[0107]

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)~~
mvp_l1_flag[x0][y0]	u(1)
} else {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
}	
…	

[0108]

[0109] 표 6은 위에서 제안된 복잡도 감소 양태들로 얻어진 성능 결과들을 도시한다. 제안된 단순화들로부터 압축 효율에서의 변화가 거의 없다는 것을 알 수 있다.

[0110] 표 6

[0111]

	랜덤 액세스 메인 10		
	Over VTM-2.0.1		
	Y	U	V
클래스 A1	0.01%	0.01%	-0.07%
클래스 A2	0.00%	0.02%	0.02%
클래스 B	0.02%	0.06%	0.06%
클래스 C	0.00%	0.01%	0.05%
클래스 E			
전체	0.01%	0.03%	0.02%
클래스 D	0.02%	0.09%	-0.02%
클래스 F (선택적)	0.04%	0.10%	0.19%

[0112] 상기에서, 예들이 VVC 드래프트 4와 관련하여 설명된다. 하기에서, 인터 서브 블록 변환, 인트라 서브 파티션(ISP) 코딩 모드, 다수 참조 라인 인트라 예측, 및 SMVD(symmetrical motion vector difference) 인터 툴의 시그널링을 포함하는 몇몇 예들이 VVC 드래프트 5와 관련하여 설명된다("Versatile Video Coding (Draft 5)", B. Bross et al., 14th JVET meeting, March 19-27, 2019, Geneva, CH 참조).

[0113] 서브 블록 변환의 시그널링

[0114] **VVC 드래프트 5에서의 서브 블록 변환(SBT)**

[0115] 0이 아닌 잔차 블록으로서 시그널링되는 인터 예측된 CU에 대해, SBT 툴은 CU를 이진 방식으로 2개의 변환 유닛(TU)으로 분할한다. 2개의 결과적인 TU 중 하나는 0이 아닌 잔차를 갖고, 다른 하나는 0 잔차 데이터만을 갖는다. 적용된 이진 분할은 대칭 또는 비대칭일 수 있다. 대칭 분할의 경우에, 2개의 결과적인 TU는 동일한 크기를 가지며, 이는 분할의 배향에서 CU의 크기의 절반이다. 비대칭 이진 분할의 경우에, 하나의 TU는 분할 배향을 따라 부모 CU의 1/4과 동일한 크기를 갖고, 다른 TU 크기는 분할 배향을 따라 CU의 크기의 3/4이다.

[0116] 공간 분할에 부가하여, 0이 아닌 잔차를 갖는 TU가 추론된 적응적 변환으로 코딩된다. 이용된 1D 변환은 도 17에 도시된 바와 같이 0이 아닌 잔차 TU의 위치에 의존하며, 여기서 부분 "A"는 0이 아닌 잔차 데이터를 갖는 TU이고, 다른 TU는 0 잔차 데이터만을 갖는다.

[0117] 고려된 코딩 유닛의 TU 분할은 3개의 플래그들을 통해 시그널링된다. 먼저, cu_sbt_flag는 고려된 CU에 대한 SBT의 이용을 나타낸다. 다음, SBT가 이용되는 경우, SBT 타입 및 SBT 위치 정보가 시그널링된다. 이것은 다음의 3개의 코딩된 플래그들의 형태를 취한다:

[0118] - cu_sbt_quad_flag는 비대칭 이진 분할의 이용을 표시한다. 이것은 현재 CU에 대해 대칭 및 비대칭 분할들 모두가 허용되는 경우에 코딩된다.

[0119] - cu_sbt_horizontal_flag는 이진 분할의 배향을 표시한다. 이것은 현재 CU에 대해, 그리고 이전에 시그널링된 SBT 분할 타입(비대칭이든 아니든)에 대해 수평 및 수직 분할들 모두가 허용되는 경우에 코딩된다.

[0120] - cu_sbt_pos_flag는 고려된 CU의 텍스처 데이터를 코딩하는데 이용된 0이 아닌 잔차 TU의 위치를 표시한다.

[0121] VVC 드래프트 5에서, 4개의 상기의 플래그들은 컨텍스트 기반 코딩된다. 이 양태에 대응하는 VVC 사양의 일부가 표 7에 도시된다.

[0122] 표 7. VVC 드래프트 5에서 컨텍스트 코딩된 빈들을 갖는 신택스 요소들에 대한 CABAC 컨텍스트들의 할당(문서 JVET-N1001의 절단된 표 9-14)

[0123]

신택스 요소	binIdx					
	0	1	2	3	4	>= 5
...	...	...	...	...	...	...
cu_sbt_flag	(cbWidth * cbHeight < 256) ? 1 : 0	na	na	na	na	na
cu_sbt_quad_flag	0	na	na	na	na	na
cu_sbt_horizontal_flag	(cbWidth = = cbHeight) ? 0 : (cbWidth < cbHeight) ? 1 : 2	na	na	na	na	na
cu_sbt_pos_flag	0	na	na	na	na	na
...	...	...	...	...	...	...

[0124] 도 18은 VVC 드래프트 5에 명시된 바와 같은 SBT 모드의 디코딩 프로세스를 도시한다. 단계 1810에서, cu_sbt_flag를 디코딩하기 위한 컨텍스트 id는 ctxId = (width*height) <= 256 ? 1:0으로서 획득된다. 그 후, 단계 1820에서, 이진 심볼 cu_sbt_flag는 컨텍스트 ctxId로 CABAC 디코딩된다. cu_sbt_flag가 0과 동일하면 (1830), 서브 블록 변환 툴은 이용되지 않는다. 그렇지 않고, cu_sbt_flag가 0이 아니면, 단계 1840에서 이진 심볼 cu_sbt_quad_flag는 컨텍스트 ctxId=0으로 CABAC 디코딩된다. 단계 1850에서, 디코더는 현재 CU에 대해 수직 및 수평 분할이 허용되는지를 체크한다. 예인 경우, 단계 1860에서, cu_sbt_horizontal_flag를 디코딩하기 위한 컨텍스트 id는 ctxId=(cuWidth == cuHeight) ? 0 : (cuWidth <cuHeight? 1: 2)로서 획득된다. 이어서, 단계 1870에서, 이진 심볼 cu_sbt_horizontal_flag가 컨텍스트 ctxId로 CABAC 디코딩된다. 단계 1880에서, 이진 심볼 cu_sbt_pos_flag는 컨텍스트 ctxId = 0으로 CABAC 디코딩된다. 프로세스는 단계 1899에서 종료한다.

[0125] **SBT 모드의 단순화된 코딩**

[0126] 제안된 실시예에 따르면, 'cu_sbt_pos_flag'는 정규(컨텍스트 기반) CABAC 모드 대신에 바이패스 모드에서 코딩된다. 실제로, 이러한 'cu_sbt_pos_flag'의 단순화된 코딩은 엔트로피 코딩을 단순화하면서 전체 코덱 코딩 효율에 무시할만한 영향을 미치는 일이 발생한다.

[0127] 다른 실시예에 따르면, 'cu_sbt_quad_flag'는 정규 모드보다는 바이패스 모드에서 코딩된다. 이러한 단순화는 또한 무시할만한 방식으로 코덱 성능에 영향을 미친다.

[0128] 다른 실시예에 따르면, 'cu_sbt_pos_flag' 및 'cu_sbt_quad_flag' 둘다는 정규 모드 대신에 바이패스 모드에서 코딩된다.

[0129] 다른 실시예에 따르면, 'cu_sbt_horizontal_flag'는 정규 모드보다는 바이패스 모드에서 코딩된다. 이러한 단순화는 또한 무시할만한 방식으로 코덱 성능에 영향을 미친다.

[0130] 다른 실시예에 따르면, 3개의 플래그들 'cu_sbt_pos_flag', 'cu_sbt_quad_flag' 및 'cu_sbt_horizontal_flag'는 정규 모드 대신에 바이패스 모드에서 코딩된다. 표 8의 시뮬레이션 결과들에 의해 보여지는 바와 같이, 이러한 전체 수정으로부터 코딩 효율의 손실은 거의 없다.

[0131] 표 8. VTM-5.0(VVC 드래프트 5 참조 소프트웨어)에 대한 코딩 효율

[0132]

	랜덤 액세스 메인 10				
	Over VTM-2.0.1				
	Y	U	V	EncT	DecT
클래스 A1	0.02%	-0.17%	0.09%	97%	102%
클래스 A2	0.01%	-0.02%	0.07%	99%	104%
클래스 B	0.02%	0.01%	-0.01%	94%	101%
클래스 C	0.01%	-0.17%	-0.13%	94%	94%
클래스 E					
전체	0.01%	-0.08%	-0.01%	96%	100%
클래스 D	0.02%	-0.12%	0.24%	97%	91%
클래스 F (선택적)	0.03%	0.02%	0.01%	98%	97%

[0133] 도 19a는 'cu_sbt_pos_flag' 및 'cu_sbt_quad_flag' 모두가 바이패스 모드에서 코딩(1940, 1980)되는 실시예에 따른 디코딩 프로세스를 도시한다. 도 19b는 3개의 플래그들 'cu_sbt_pos_flag', 'cu_sbt_quad_flag' 및 'cu_sbt_horizontal_flag'가 바이패스 모드에서 코딩되는 실시예에 따른 디코딩 프로세스를 도시한다.

[0134] 표 9. VVC 드래프트 5에 대한 수정

[0135]

신택스 요소	binIdx					
	0	1	2	3	4	>= 5
...	...	...	...	...	...	...
cu_sbt_flag	(cbWidth * cbHeight < 256) ? 1 : 0	na	na	na	na	na
cu_sbt_quad_flag	*바이패스*	na	na	na	na	na
cu_sbt_horizontal_flag	(cbWidth = = cbHeight) ? 0 : (cbWidth < cbHeight) ? 1 : 2	na	na	na	na	na
cu_sbt_pos_flag	*바이패스*	na	na	na	na	na
...	...	...	...	...	...	...

[0136] VVC 드래프트 5의 규범적 신택스 사양은 표 10에 도시된 바와 같이 수정될 수 있다. 특히, 신택스 요소들 cu_sbt_quad_flag 및 cu_sbt_pos_flag에 대한 서술자는 ae(v)로부터 u(1)로 변경된다.

[0137] 표 10. VVC 드래프트 5에 명시된 코딩 유닛 신택스에 대한 수정

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	서술자
…	
if(cu_cbf) {	
if(CuPredMode[x0][y0] = = MODE_INTER && sps_sbt_enabled_flag && !ciip_flag[x0][y0] && !MergeTriangleFlag[x0][y0]) {	
if(cbWidth <= MaxSbtSize && cbHeight <= MaxSbtSize) {	
allowSbtVerH = cbWidth >= 8	
allowSbtVerQ = cbWidth >= 16	
allowSbtHorH = cbHeight >= 8	
allowSbtHorQ = cbHeight >= 16	
if(allowSbtVerH || allowSbtHorH || allowSbtVerQ || allowSbtHorQ)	
cu_sbt_flag	ae(v)
}	
if(cu_sbt_flag) {	
if((allowSbtVerH || allowSbtHorH) && (allowSbtVerQ || allowSbtHorQ))	
~~**cu_sbt_quad_flag**~~	~~ae(v)~~
cu_sbt_quad_flag	u(1)
if((cu_sbt_quad_flag && allowSbtVerQ && allowSbtHorQ) || (!cu_sbt_quad_flag && allowSbtVerH && allowSbtHorH))	
cu_sbt_horizontal_flag	ae(v)
~~**cu_sbt_pos_flag**~~	~~ae(v)~~
cu_sbt_pos_flag	u(1)
}	
}	
transform_tree(x0, y0, cbWidth, cbHeight, treeType)	
}	
}	
}	

[0138]

[0139] 인트라 서브 파티션 코딩 모드의 시그널링

[0140] **VVC 드래프트 5에서의 인트라 서브 파티션**

[0141] VVC 드래프트 5에서 명시된 인트라 서브 파티션(ISP) 코딩 모드는 INTRA CU를 2개 또는 4개의 서브 파티션들로 수평으로 또는 수직으로 분할할 수 있다. 분할은 표 11에 도시된 바와 같이 블록 크기에 의존한다. 기본적으로, 4x4 CU는 추가로 분할될 수 없다. 4x8 또는 8x4 크기의 CU는 2개의 Tu들(즉, 서브 파티션들)로 분할된다. 다른 CU들은 4개의 Tu들로 분할된다.

[0142] 표 11. CU 크기에 따른 ISP 서브 파티션들의 수

블록 크기	서브 파티션들의 수
4 × 4	분할되지 않음
4 × 8 및 8 × 4	2
모든 다른 경우들	4

[0143]

[0144] 도 20a는 4x8 및 8x4 INTRA 코딩 유닛들에 대한 허용된 ISP 분할들을 도시하고, 도 20b는 4x8 또는 8x4와 상이한 크기를 갖는 INTRA 코딩 유닛들에 대한 허용된 ISP 분할들을 도시한다. ISP 모드에서 코딩된 CU 내에서, TU들은 순차적으로 디코딩되고, CU 레벨 상에서 시그널링되는 동일한 인트라 예측 모드를 이용하여 TU로부터 TU로 인트라 예측된다. 마지막으로, 잔차 코딩은 또한 인트라 서브 파티션들 크기에 따라 적응된다. 실제로, 서브 파티션들은 크기 1xN, Nx1, 2xN 또는 Nx2일 수 있고, 크기 1x16, 1x1, 2x16 또는 16x2의 코딩 그룹이 이들 각각의 경우에 이용된다.

[0145] ISP 코딩 모드는 VVC 드래프트 5에서 2개의 연속적인 플래그들을 통해 시그널링된다.

[0146] - intra_subpartitions_mode_flag는 주어진 인트라 CU에 대한 ISP 모드의 이용을 표시한다.

[0147] - intra_subpartitions_split_flag는 인트라 서브 파티션으로의 분할의 배향을 표시한다.

[0148] 상기의 2개의 플래그들은, 다음과 같이, 고려된 CU와 연관된 "IntraSubPartitionsSplitType"의 값을 디코딩하는데 이용된다:

[0149] $$IntraSubPartitionsSplitType = intra_subpartitions_mode_flag == 0 \ ? \ 0 : (1+ intra_subpartitions_split_flag)$$

[0150] "IntraSubPartitionsSplitType" 값의 의미가 표 12에 도시되어 있다.

[0151] 표 12. VVC 드래프트 5에 명시된 바와 같은 IntraSubPartitionsSplitType의 의미

[0152]

IntraSubPartitionsSplitType	IntraSubPartitionsSplitType의 명칭
0	ISP_NO_SPLIT
1	ISP_HOR_SPLIT
2	ISP_VER_SPLIT

[0153] 이들 2개의 플래그들은 표 13에 나타낸 바와 같이 VVC 드래프트 5에 따라 컨텍스트 기반 코딩된다.

[0154] 표 13. ISP 관련 신택스 요소들을 위해 이용된 엔트로피 코딩 모드를 나타내는, VVC 드래프트 5로부터의 절단된 표

[0155]

신택스 요소	binIdx					
	0	1	2	3	4	>= 5
...	...	...	...	...	...	...
intra_subpartitions_mode_flag	0	na	na	na	na	na
intra_subpartitions_split_flag	0	na	na	na	na	na
...	...	...	...	...	...	...

[0156] **ISP 모드의 단순화된 코딩**

[0157] 일 실시예에서, 빈 'intra_subpartitions_split_flag', 즉, CU의 ISP 모드를 코딩하는데 이용된 제2 플래그는 바이패스 모드에서 코딩된다. 이 실시예와 연관된 VVC 드래프트 5 사양에 대한 수정이 표 14 및 표 15에 도시된다. 특히, 신택스 요소 'intra_subpartitions_split_flag'에 대한 서술자는 ae(v)로부터 u(1)로 변경된다.

[0158] 표 14. 신택스 요소들에 대한 CABAC 컨텍스트 할당을 명시하는 VVC 드래프트 5에 대한 수정

[0159]

신택스 요소	binIdx					
	0	1	2	3	4	>= 5
...	...	...	...	...	...	...
intra_subpartitions_mode_flag	0	na	na	na	na	na
intra_subpartitions_split_flag	*바이패스*	na	na	na	na	na
...	...	...	...	...	...	...

[0160] 표 15. VVC 드래프트 5에서의 코딩 유닛 신택스에 대한 수정

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	서술자
…	
if(CuPredMode[x0][y0] = = MODE_INTRA) {	
if(sps_pcm_enabled_flag && cbWidth >= MinIpcmCbSizeY && cbWidth <= MaxIpcmCbSizeY && cbHeight >= MinIpcmCbSizeY && cbHeight <= MaxIpcmCbSizeY)	
pcm_flag[x0][y0]	ae(v)
if(pcm_flag[x0][y0]) {	
while(!byte_aligned( ))	
pcm_alignment_zero_bit	f(1)
pcm_sample(cbWidth, cbHeight, treeType)	
} else {	
if(treeType = = SINGLE_TREE || treeType = = DUAL_TREE_LUMA) {	
if((y0 % CtbSizeY) > 0)	
intra_luma_ref_idx[x0][y0]	ae(v)

[0161]

if (intra_luma_ref_idx[x0][y0] = = 0 && (cbWidth <= MaxTbSizeY || cbHeight <= MaxTbSizeY) && (cbWidth * cbHeight > MinTbSizeY * MinTbSizeY))	
intra_subpartitions_mode_flag[x0][y0]	ae(v)
if(intra_subpartitions_mode_flag[x0][y0] = = 1 && cbWidth <= MaxTbSizeY && cbHeight <= MaxTbSizeY)	
~~**intra_subpartitions_split_flag**[x0][y0]~~	~~ae(v)~~
intra_subpartitions_split_flag[x0][y0]	u(1)
if(intra_luma_ref_idx[x0][y0] = = 0 && intra_subpartitions_mode_flag[x0][y0] = = 0)	
intra_luma_mpm_flag[x0][y0]	ae(v)
if(intra_luma_mpm_flag[x0][y0])	
intra_luma_mpm_idx[x0][y0]	ae(v)
else	
intra_luma_mpm_remainder[x0][y0]	ae(v)
}	
if(treeType = = SINGLE_TREE || treeType = = DUAL_TREE_CHROMA)	
intra_chroma_pred_mode[x0][y0]	ae(v)
}	
…	

[0162]

[0163] 다수 참조 라인 인트라 예측의 시그널링

[0164] **VVC 드래프트 5에서의 다수 참조 라인 인트라 예측**

[0165] VVC 드래프트 5에서 INTRA CU들에서 이용된 다수 참조 라인 인트라 예측은, 3개의 참조 라인들 중에서 선택된, 현재 CU의 상부 및 좌측의 하나의 참조 라인 및 열에 속하는 재구성된 참조 샘플들에 기초하여 루마 블록의 각도 인트라 예측을 수행한다. 인트라 예측을 위해 이용된 참조 라인은 신택스 요소 'intra_luma_ref_idx'를 통해 비트스트림에서 시그널링된다. 각각의 참조 라인은, 도 21에 도시된 바와 같이, 그의 인덱스에 의해 식별된다. VVC 드래프트 5에서 실제로 이용된 참조는 도 21에 도시된 라인들 0, 1 및 3이다. 이 신택스 요소는 인트라 예측 모드 이전에 시그널링된다. 참조 라인이 라인 0과 상이한 경우, 즉, CU에 가장 가까운 라인.

[0166] 'intra_luma_ref_idx'는 다음과 같이 코딩된다. 이것은 절단 라이스 빈 스트링으로서 이진화된다. 이는, 그것이 1과 동일한 일련의 정규 CABAC 빈들에 의해 코딩되고, 0과 동일한 정규 CABAC 빈에 의해 종단된다는 것을 의미한다. 전반적으로, 최대 3개의 빈들이 시그널링된다.

[0167] VVC 드래프트 5에 따른 'intra_luma_ref_idx'의 디코딩 프로세스가 도 22에 도시되어 있다. 도 22에서, 최대 4개의 참조 라인들을 수반하는 일반적인 프로세스가 이용된다. 어레이 lineIDx[.]는 4개의 라인 참조 인덱스들로 이루어지고, 값 'MaxNumRefLInes'는 인트라 예측을 위한 허용된 참조 라인들의 최대 수를 나타낸다. 'MaxNumRefLInes'는 VVC에 대해 3과 동일하고, 어레이 lineIdx는 다음의 요소들로 이루어진다:

[0168] lineIdx = {0, 1, 3}.

[0169] 신택스 요소 'intra_luma_ref_idx'의 VVC 디코딩 프로세스는 다음과 같이 진행한다. 프로세스의 출력은 디코딩된 multiRefIdx이다. 특히, 단계 2210에서, 현재 CU에 대해 다수 참조 라인들이 허용되는지를 체크한다. 예인 경우, 출력 값 multiRefIdx는 0으로 초기화(2220)된다. MaxNumRefLines가 1보다 높지 않은 경우(2225), 프로세스는 종료된다. 그렇지 않으면, 단계 2230에서, CABAC 정규 빈은 인덱스 0을 갖는 단일 CABAC 컨텍스트로 파싱된다. 0과 동일하면, multiRefIdx 값은 변경되지 않고 프로세스는 종료된다. 그렇지 않으면, multiRefIdx는 lineIdx[1]과 동일하게 설정된다.

[0170] 단계 2255에서, MaxNumRefLines가 2보다 엄밀히 높은 경우(2250), 제2 정규 CABAC 빈은 식별자 1을 갖는 단일 컨텍스트로 디코딩된다. 디코딩된 빈이 0이면, multiRefIdx는 변경되지 않고 프로세스는 종료된다. 그렇지 않으면, mutiRefIdx는 lineIdx[2]와 동일하게 설정된다(2260). MaxNumRefLines가 3보다 엄밀히 높은 경우(2270), 제3 정규 CABAC 빈은 식별자 2를 갖는 단일 컨텍스트로 디코딩(2280)된다. 디코딩된 빈이 0이면, multiRefIdx는 변경되지 않고 프로세스는 종료(2299)된다. 그렇지 않으면, mutiRefIdx는 lineIdx[3]과 동일하게 설정(2290)된다. 실제로 이 단계는 VVC 드래프트 5 사양에 따라 발생되지 않는데, 그 이유는 VVC 드래프트 5에 대해 선택된 설계에서, 최대 3개의 참조 라인들이 전술한 바와 같이 이용될 수 있기 때문이다. 따라서, 조건부 "MaxNumRefLines>3"은 VVC 드래프트 5의 범위에서 항상 거짓이다.

[0171] 이미 설명되고 표 16에 도시된 바와 같이, 2개의 정규 코딩된 빈들은 VVC 드래프트 5에서 intra_luma_ref_idx 신택스 요소를 시그널링하는데 이용되며, 각각의 빈은 단일 CABAC 컨텍스트를 이용한다.

[0172] 표 16. VVC 드래프트 5에서와 같이, 'intra_luma_ref_idx'의 컨텍스트 기반 코딩된 빈들에 대한 CABAC 컨텍스트 할당

신택스 요소	binIdx					
	0	1	2	3	4	>= 5
...	...	...	...	...	...	...
intra_luma_ref_idx[ ][ ]	0	1	na	na	na	na
...	...	...	...	...	...	...

[0173]

[0174] **다수 참조 라인 인덱스의 단순화된 코딩**

[0175] 일 실시예에서, 'intra_luma_ref_idx' 신택스 요소의 코딩은 단순화되고, 이 신택스 요소의 제1 빈만이 정규 모드에서 코딩된다. 이 신택스 요소에 대한 제안된 수정된 파싱 프로세스가 도 23에 도시되어 있다. 특히, 단계 2355 및 2380에서, 제2 빈 및 제3 빈은 바이패스 모드에서 디코딩된다. 다른 단계들은 도 22에 도시된 것들과 유사하다.

[0176] 이 실시예에 따르면, VVC 드래프트 5 사양은 표 17에 도시된 바와 같이 수정될 수 있다.

[0177] 표 17. 'intra_luma_ref_idx'의 컨텍스트 기반 코딩된 빈들에 대한 CABAC 컨텍스트 할당

신택스 요소	binIdx					
	0	1	2	3	4	>= 5
...	...	...	...	...	...	...
intra_luma_ref_idx[ ][ ]	0	*바이패스*	na	na	na	na
...	...	...	...	...	...	...

[0178]

[0179] SMVD 플래그의 시그널링

[0180] **VVC 드래프트 5에서의 SMVD 플래그**

[0181] sym_mvd_flag 신택스 요소는 INTER 코딩 유닛에 대한 대칭 모션 벡터 차이의 이용을 나타낸다. VVC의 SMVD 모션 벡터 코딩 프로세스는 제1 참조 픽처 리스트 L0에 관한 주어진 CU의 모션 벡터 차이를 코딩한다. 이어서, 다른 참조 픽처 리스트(L1)에서의 고려된 CU의 모션 벡터 차이가 L0 모션 벡터 차이로부터 추론된다. 실제로, 이 모드에서, 2개의 모션 벡터 차이들은 대칭이다. L1 모션 벡터 차이는 x 성분 및 y 성분 둘다에서 L0 모션 벡터 차이의 반대와 같다. VVC 드래프트 5에서, 이 sym_mvd_flag는 단일 CABAC 컨텍스트를 갖는 정규 CABAC 모드에서 코딩되고 디코딩된다.

[0182] **SMVD 플래그의 단순화된 코딩**

[0183] 일 실시예에서, 바이패스 모드에서 이 플래그를 인코딩 및 디코딩하는 것이 제안된다. 제안된 단순화는 VVC의 압축 효율에 영향을 미치지 않는다. 이 실시예에 따르면, VVC 드래프트 5 신택스 사양은 표 18에 도시된 바와 같이 수정될 수 있다. 특히, svm_mvd_flag에 대한 서술자는 ae(v)로부터 u(1)로 변경된다.

[0184] 표 18. VVC 드래프트 5를 통한 smvd_flag의 단순화된 시그널링

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	서술자
…	
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(sps_smvd_enabled_flag && inter_pred_idc[x0][y0] = = PRED_BI && !inter_affine_flag[x0][y0] && RefIdxSymL0 > −1 && RefIdxSymL1 > −1)	
~~**sym_mvd_flag**[x0][y0]~~	~~ae(v)~~
sym_mvd_flag[x0][y0]	u(1)
if(inter_pred_idc[x0][y0] != PRED_L1) {	
if(NumRefIdxActive[0] > 1 && !sym_mvd_flag[x0][y0])	
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	
}	
…	

[0185]

[0186] 다양한 방법들이 본 명세서에서 설명되었고, 방법들 각각은 설명된 방법을 달성하기 위한 하나 이상의 단계 또는 동작을 포함한다. 방법의 적절한 동작을 위해 단계들 또는 동작들의 특정 순서가 요구되지 않는 한, 특정 단계들 및/또는 동작들의 순서 및/또는 이용은 수정되거나 결합될 수 있다.

[0187] 본 출원에서 설명된 다양한 방법들 및 다른 양태들은 도 1 및 도 4에 도시된 바와 같은 비디오 인코더(100) 및 디코더(200)의 모듈들, 예를 들어, 엔트로피 인코딩 및 디코딩 모듈들(145, 230)을 수정하기 위해 이용될 수 있다. 또한, 본 양태들은 VVC 또는 HEVC로 제한되지 않고, 예를 들어, 다른 표준들 및 권고들, 및 임의의 그러한 표준들 및 권고들의 확장들에 적용될 수 있다. 달리 표시되지 않거나 또는 기술적으로 배제되지 않는 한, 본 출원에서 설명되는 양태들은 개별적으로 또는 조합하여 이용될 수 있다.

[0188] 다양한 수치 값들, 예를 들어, 컨텍스트 모델 ID들이 본 출원에서 이용된다. 특정 값들은 예시적인 목적들이고, 설명된 양태들은 이러한 특정 값들로 제한되지 않는다.

[0189] 도 24는 다양한 양태들 및 실시예들이 구현되는 시스템의 예의 블록도를 도시한다. 시스템(2400)은 아래에 설명된 다양한 컴포넌트들을 포함하는 디바이스로서 구현될 수 있고, 본 출원에 설명된 양태들 중 하나 이상을 수행하도록 구성된다. 그러한 디바이스들의 예는, 제한적인 것은 아니지만, 개인용 컴퓨터들, 랩톱 컴퓨터들, 스마트폰들, 태블릿 컴퓨터들, 디지털 멀티미디어 셋톱 박스들, 디지털 텔레비전 수신기들, 개인 비디오 기록 시스템들, 접속된 가전 기기들, 및 서버들과 같은 다양한 전자 디바이스들을 포함한다. 시스템(2400)의 요소들은, 단독으로 또는 조합하여, 단일의 집적 회로, 다수의 IC들, 및/또는 개별 컴포넌트들로 구현될 수 있다. 예를 들어, 적어도 하나의 실시예에서, 시스템(2400)의 처리 및 인코더/디코더 요소들은 다수의 IC들 및/또는 개별 컴포넌트들에 걸쳐 분산된다. 다양한 실시예들에서, 시스템(2400)은, 예를 들어, 통신 버스를 통해 또는 전용 입력 및/또는 출력 포트들을 통해, 다른 시스템들 또는 다른 전자 디바이스들에 통신가능하게 결합된

다. 다양한 실시예들에서, 시스템(2400)은 본 출원에서 설명된 양태들 중 하나 이상을 구현하도록 구성된다.

[0190] 시스템(2400)은, 예를 들어, 본 출원에서 설명된 다양한 양태들을 구현하기 위해 그 안에 로딩된 명령어들을 실행하도록 구성된 적어도 하나의 프로세서(2410)를 포함한다. 프로세서(2410)는 내장형 메모리, 입력 출력 인터페이스, 및 본 기술분야에 알려진 바와 같은 다양한 다른 회로들을 포함할 수 있다. 시스템(2400)은 적어도 하나의 메모리(2420)(예를 들어, 휘발성 메모리 디바이스, 및/또는 비휘발성 메모리 디바이스)를 포함한다. 시스템(2400)은, 제한적인 것은 아니지만, EEPROM, ROM, PROM, RAM, DRAM, SRAM, 플래시, 자기 디스크 드라이브, 및/또는 광학 디스크 드라이브를 포함하는 비휘발성 메모리 및/또는 휘발성 메모리를 포함할 수 있는 저장 디바이스(2440)를 포함한다. 저장 디바이스(2440)는, 비제한적인 예들로서, 내부 저장 디바이스, 부착된 저장 디바이스, 및/또는 네트워크 액세스가능 저장 디바이스를 포함할 수 있다.

[0191] 시스템(2400)은, 예를 들어, 인코딩된 비디오 또는 디코딩된 비디오를 제공하기 위해 데이터를 처리하도록 구성된 인코더/디코더 모듈(2430)을 포함하고, 인코더/디코더 모듈(2430)은 그 자신의 프로세서 및 메모리를 포함할 수 있다. 인코더/디코더 모듈(2430)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 둘다를 포함할 수 있다. 추가적으로, 인코더/디코더 모듈(2430)은 시스템(2400)의 개별 요소로서 구현될 수 있거나, 본 기술분야의 통상의 기술자에게 알려진 바와 같이 하드웨어와 소프트웨어의 조합으로서 프로세서(2410) 내에 포함될 수 있다.

[0192] 본 출원에서 설명된 다양한 양태들을 수행하기 위해 프로세서(2410) 또는 인코더/디코더(2430) 상에 로딩될 프로그램 코드는 저장 디바이스(2440)에 저장되고, 프로세서(2410)에 의한 실행을 위해 메모리(2420) 상에 후속하여 로딩될 수 있다. 다양한 실시예들에 따르면, 프로세서(2410), 메모리(2420), 저장 디바이스(2440), 및 인코더/디코더 모듈(2430) 중 하나 이상은 본 출원에서 설명된 프로세스들의 수행 동안 다양한 항목들 중 하나 이상을 저장할 수 있다. 그러한 저장된 항목들은, 제한적인 것은 아니지만, 입력 비디오, 디코딩된 비디오 또는 디코딩된 비디오의 부분들, 비트스트림, 행렬들, 변수들, 및 방정식들, 공식들, 연산들 및 연산 로직의 처리로부터의 중간 또는 최종 결과들을 포함할 수 있다.

[0193] 몇몇 실시예들에서, 프로세서(2410) 및/또는 인코더/디코더 모듈(2430) 내부의 메모리는 명령어들을 저장하고, 인코딩 또는 디코딩 동안 필요한 처리를 위한 작업 메모리를 제공하는데 이용된다. 그러나, 다른 실시예들에서, 처리 디바이스(예를 들어, 처리 디바이스는 프로세서(2410) 또는 인코더/디코더 모듈(2430) 중 어느 하나일 수 있음) 외부의 메모리는 이러한 기능들 중 하나 이상을 위해 이용된다. 외부 메모리는 메모리(2420) 및/또는 저장 디바이스(2440), 예를 들어, 동적 휘발성 메모리 및/또는 비휘발성 플래시 메모리일 수 있다. 몇몇 실시예들에서, 외부 비휘발성 플래시 메모리는 텔레비전의 운영 체제를 저장하기 위해 이용된다. 적어도 일 실시예에서, RAM과 같은 고속 외부 동적 휘발성 메모리는 MPEG-2, HEVC 또는 VVC와 같은 비디오 코딩 및 디코딩 동작들을 위한 작업 메모리로서 이용된다.

[0194] 시스템(2400)의 요소에 대한 입력은 블록(2405)에 나타낸 바와 같이 다양한 입력 디바이스들을 통해 제공될 수 있다. 그러한 입력 디바이스들은, 제한적인 것은 아니지만, (i) 예를 들어, 브로드캐스터에 의해 오버 더 에어(over the air)로 송신된 RF 신호를 수신하는 RF 부분, (ii) 복합 입력 단자, (iii) USB 입력 단자, 및/또는 (iv) HDMI 입력 단자를 포함한다.

[0195] 다양한 실시예들에서, 블록(2405)의 입력 디바이스들은 본 기술분야에 알려진 바와 같은 연관된 개개의 입력 처리 요소들을 갖는다. 예를 들어, RF 부분은 (i) 원하는 주파수를 선택하는 것(신호를 선택하는 것, 또는 신호를 주파수들의 대역으로 대역 제한하는 것으로도 지칭됨), (ii) 선택된 신호를 하향 변환하는 것, (iii) (예를 들어) 특정 실시예들에서 채널로 지칭될 수 있는 신호 주파수 대역을 선택하기 위해 주파수들의 더 좁은 대역으로 다시 대역 제한하는 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 적합한 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, 하향 변환기들, 복조기들, 에러 정정기들, 및 역다중화기들을 포함한다. RF 부분은, 예를 들어, 수신된 신호를 더 낮은 주파수(예를 들어, 중간 주파수 또는 근-기저대역 주파수)로 또는 기저대역으로 하향 변환하는 것을 포함하는 다양한 이러한 기능들을 수행하는 튜너를 포함할 수 있다. 하나의 셋톱 박스 실시예에서, RF 부분 및 그의 연관된 입력 처리 요소는 유선(예를 들어, 케이블) 매체를 통해 송신된 RF 신호를 수신하고, 필터링, 하향 변환, 및 원하는 주파수 대역으로의 다시 필터링하는 것에 의해 주파수 선택을 수행한다. 다양한 실시예들은 전술한 (및 다른) 요소들의 순서를 재배열하고,

이 요소들의 일부를 제거하고/하거나 유사하거나 상이한 기능들을 수행하는 다른 요소들을 추가한다. 요소들을 추가하는 것은 기존 요소들 사이에 요소들을 삽입하는 것, 예를 들어, 증폭기들 및 아날로그-디지털 변환기를 삽입하는 것을 포함할 수 있다. 다양한 실시예들에서, RF 부분은 안테나를 포함한다.

[0196] 추가적으로, USB 및/또는 HDMI 단말들은 시스템(2400)을 USB 및/또는 HDMI 접속들을 거쳐 다른 전자 디바이스들에 접속하기 위한 개개의 인터페이스 프로세서들을 포함할 수 있다. 입력 처리의 다양한 양태들, 예를 들어, 리드-솔로몬 에러 정정(Reed-Solomon error correction)이, 예를 들어, 필요에 따라 별개의 입력 처리 IC 내에서 또는 프로세서(2410) 내에서 구현될 수 있다는 것을 이해해야 한다. 유사하게, USB 또는 HDMI 인터페이스 처리의 양태들은 필요에 따라 별개의 인터페이스 IC들 내에서 또는 프로세서(2410) 내에서 구현될 수 있다. 복조된, 에러 정정된, 및 역다중화된 스트림은, 예를 들어, 프로세서(2410), 및 출력 디바이스 상에 제시하기 위해 필요에 따라 데이터스트림을 처리하기 위해 메모리 및 저장 요소들과 결합하여 동작하는 인코더/디코더(2430)를 포함하는 다양한 처리 요소들에 제공된다.

[0197] 시스템(2400)의 다양한 요소들은 통합된 하우징 내에 제공될 수 있고, 통합된 하우징 내에서, 다양한 요소들은 적절한 접속 배열(2415), 예를 들어, I2C 버스, 배선, 및 인쇄 회로 보드들을 포함하는, 본 기술분야에 알려진 바와 같은 내부 버스를 이용하여 상호접속되고 그들 사이에 데이터를 송신할 수 있다.

[0198] 시스템(2400)은 통신 채널(2490)을 통해 다른 디바이스들과의 통신을 가능하게 하는 통신 인터페이스(2450)를 포함한다. 통신 인터페이스(2450)는, 제한적인 것은 아니지만, 통신 채널(2490)을 통해 데이터를 송신 및 수신하도록 구성된 트랜시버를 포함할 수 있다. 통신 인터페이스(2450)는, 제한적인 것은 아니지만, 모뎀 또는 네트워크 카드를 포함할 수 있고, 통신 채널(2490)은, 예를 들어, 유선 및/또는 무선 매체 내에서 구현될 수 있다.

[0199] 다양한 실시예들에서, 데이터는 IEEE 802.11과 같은 Wi-Fi 네트워크를 이용하여 시스템(2400)에 스트리밍된다. 이러한 실시예들의 Wi-Fi 신호는 Wi-Fi 통신들을 위해 적응되는 통신 채널(2490) 및 통신 인터페이스(2450)를 통해 수신된다. 이러한 실시예들의 통신 채널(2490)은 전형적으로 스트리밍 애플리케이션들 및 다른 오버-더-톱(over-the-top) 통신들을 허용하기 위해 인터넷을 포함하는 외부 네트워크들에 대한 액세스를 제공하는 액세스 포인트 또는 라우터에 접속된다. 다른 실시예들은 입력 블록(2405)의 HDMI 접속을 통해 데이터를 전달하는 셋톱 박스를 이용하여 스트리밍된 데이터를 시스템(2400)에 제공한다. 또 다른 실시예들은 입력 블록(2405)의 RF 접속을 이용하여 스트리밍된 데이터를 시스템(2400)에 제공한다.

[0200] 시스템(2400)은 디스플레이(2465), 스피커들(2475), 및 다른 주변 디바이스들(2485)을 포함하는 다양한 출력 디바이스들에 출력 신호를 제공할 수 있다. 다른 주변 디바이스들(2485)은, 실시예들의 다양한 예들에서, 독립형 DVR, 디스크 플레이어, 스테레오 시스템, 조명 시스템, 및 시스템(2400)의 출력에 기초하여 기능을 제공하는 다른 디바이스들 중 하나 이상을 포함한다. 다양한 실시예들에서, 제어 신호들은 AV.Link, CEC, 또는 사용자 개입으로 또는 사용자 개입 없이 디바이스-대-디바이스 제어를 가능하게 하는 다른 통신 프로토콜들과 같은 시그널링을 이용하여 시스템(2400)과 디스플레이(2465), 스피커들(2475), 또는 다른 주변 디바이스들(2485) 사이에서 통신된다. 출력 디바이스들은 개개의 인터페이스들(2460, 2470, 및 2480)을 통한 전용 접속들을 통해 시스템(2400)에 통신가능하게 결합될 수 있다. 대안적으로, 출력 디바이스들은 통신 인터페이스(2450)를 통해 통신 채널(2490)을 이용하여 시스템(2400)에 접속될 수 있다. 디스플레이(2465) 및 스피커(2475)는 전자 디바이스, 예를 들어, 텔레비전에서 시스템(2400)의 다른 컴포넌트들과 단일 유닛으로 통합될 수 있다. 다양한 실시예들에서, 디스플레이 인터페이스(2460)는 디스플레이 드라이버, 예를 들어, 타이밍 제어기(T Con) 칩을 포함한다.

[0201] 디스플레이(2465) 및 스피커(2475)는, 대안적으로, 예를 들어, 입력(2405)의 RF 부분이 별도의 셋톱 박스의 일부인 경우, 다른 컴포넌트들 중 하나 이상으로부터 분리될 수 있다. 디스플레이(2465) 및 스피커(2475)가 외부 컴포넌트들인 다양한 실시예들에서, 출력 신호는, 예를 들어, HDMI 포트들, USB 포트들, 또는 COMP 출력들을 포함하는 전용 출력 접속들을 통해 제공될 수 있다.

[0202] 실시예에 따르면, 비디오 디코딩의 방법으로서, 비트스트림으로부터 복수의 이진 심볼들을 디코딩하는 단계 – 상기 복수의 이진 심볼들 중 제1 이진 심볼은 엔트로피 디코딩 엔진의 바이패스 모드를 이용하여 디코딩됨 –; 및 이진화 방식에 기초하여, 상기 복수의 이진 심볼들에 응답하여 신택스 요소에 대한 값을 생성하는 단계를 포함하는 방법이 제공된다.

[0203] 실시예에 따르면, 비디오 인코딩의 방법으로서, 신택스 요소의 값을 나타내는 복수의 이진 심볼들에 액세스하는 단계; 및 상기 복수의 이진 심볼들을 인코딩하는 단계 – 상기 복수의 이진 심볼들 중 제1 이진 심볼은 엔트로

피 인코딩 엔진의 바이패스 모드를 이용하여 인코딩됨 – 를 포함하는 방법이 제공된다.

[0204] 다른 실시예에 따르면, 하나 이상의 프로세서를 포함하는 비디오 디코딩을 위한 장치로서, 상기 하나 이상의 프로세서는, 비트스트림으로부터 복수의 이진 심볼들을 디코딩하고 – 상기 복수의 이진 심볼들 중 제1 이진 심볼은 엔트로피 디코딩 엔진의 바이패스 모드를 이용하여 디코딩됨 –; 이진화 방식에 기초하여, 상기 복수의 이진 심볼들에 응답하여 신택스 요소에 대한 값을 생성하도록 구성되는 장치가 제공된다. 장치는 상기 하나 이상의 프로세서에 결합된 하나 이상의 메모리를 더 포함할 수 있다.

[0205] 다른 실시예에 따르면, 하나 이상의 프로세서를 포함하는 비디오 인코딩을 위한 장치로서, 상기 하나 이상의 프로세서는, 신택스 요소의 값을 나타내는 복수의 이진 심볼들에 액세스하고; 상기 복수의 이진 심볼들을 인코딩 – 상기 복수의 이진 심볼들 중 제1 이진 심볼은 엔트로피 인코딩 엔진의 바이패스 모드를 이용하여 인코딩됨 – 하도록 구성되는 장치가 제공된다.

[0206] 다른 실시예에 따르면, 비디오 디코딩의 장치로서, 비트스트림으로부터 복수의 이진 심볼들을 디코딩하는 수단 – 상기 복수의 이진 심볼들 중 제1 이진 심볼은 엔트로피 디코딩 엔진의 바이패스 모드를 이용하여 디코딩됨 –; 및 이진화 방식에 기초하여, 상기 복수의 이진 심볼들에 응답하여 신택스 요소에 대한 값을 생성하는 수단을 포함하는 장치가 제공된다.

[0207] 다른 실시예에 따르면, 비디오 인코딩의 장치로서, 신택스 요소의 값을 나타내는 복수의 이진 심볼들에 액세스하는 수단; 및 상기 복수의 이진 심볼들을 인코딩하는 수단 – 상기 복수의 이진 심볼들 중 제1 이진 심볼은 엔트로피 인코딩 엔진의 바이패스 모드를 이용하여 인코딩됨 – 를 포함하는 장치가 제공된다.

[0208] 다른 실시예에 따르면, 인코딩된 비디오를 포함하는 신호는 신택스 요소의 값을 나타내는 복수의 이진 심볼들에 액세스하는 것; 및 상기 복수의 이진 심볼들을 인코딩하는 것 – 상기 복수의 이진 심볼들 중 제1 이진 심볼은 엔트로피 디코딩 엔진의 바이패스 모드를 이용하여 인코딩됨 – 을 수행함으로써 형성된다.

[0209] 실시예에 따르면, 상기 복수의 이진 심볼들 중 하나 이상의 다른 이진 심볼은 상기 바이패스 모드를 이용하여 디코딩 또는 인코딩된다.

[0210] 실시예에 따르면, 상기 복수의 이진 심볼들의 나머지는 컨텍스트 기반 디코딩 또는 인코딩된다. 상기 복수의 이진 심볼들의 나머지의 각각의 이진 심볼은 상이한 컨텍스트 모델을 이용할 수 있다. 다른 실시예에서, 상기 복수의 이진 심볼들의 모든 이진 심볼들은 상기 바이패스 모드를 이용하여 디코딩 또는 인코딩된다.

[0211] 실시예에 따르면, 이진 플래그는 컨텍스트 기반 인코딩 또는 디코딩되고, 상기 신택스 요소의 상기 값은 상기 이진 플래그에 추가로 응답하여 생성된다.

[0212] 일 실시예에서, 상기 플래그는 상기 블록의 2개의 시간적 예측기들의 상기 가중 평균을 생성 시에 동일한 가중치가 적용되는지를 표시한다.

[0213] 실시예에 따르면, 상기 신택스 요소는 블록의 2개의 시간적 예측기들의 가중 평균을 생성 시에 이용된 가중치의 인덱스를 표시한다.

[0214] 일 실시예에서, 상기 신택스 요소는 블록의 모션 벡터를 인코딩 또는 디코딩하기 위해 어느 모션 벡터 예측기가 이용될지를 표시한다.

[0215] 일 실시예에서, SMVD(symmetrical motion vector difference)가 상기 블록에 적용되는지가 결정되고, 상기 바이패스 모드는 SMVD가 상기 블록에 적용되는 경우에만 이용된다.

[0216] 일 실시예에서, 절단 라이스 이진화가 이진화 방식으로서 이용된다.

[0217] 일 실시예에서, 서브 블록 변환이 이용될 때 현재 코딩 유닛의 텍스처 데이터를 코딩하는데 이용된 0이 아닌 잔차 변환 유닛의 위치를 표시하는 신택스 요소는 바이패스 모드에서 인코딩 및 디코딩된다.

[0218] 일 실시예에서, 비대칭 이진 분할이 서브 블록 변환에서 이용되는지를 표시하는 신택스 요소는 바이패스 모드에서 인코딩 및 디코딩된다.

[0219] 일 실시예에서, 서브 블록 변환에서 이용된 이진 분할의 배향을 표시하는 신택스 요소는 바이패스 모드에서 인코딩 및 디코딩된다.

[0220] 일 실시예에서, 현재 코딩 유닛을 인트라 서브 파티션으로 분할하는 배향을 표시하는 신택스 요소는 바이패스

모드에서 인코딩 및 디코딩된다.

[0221] 일 실시예에서, 어느 참조 라인이 인트라 예측을 위해 이용되는지를 표시하는 신택스 요소를 표현하기 위해 이용된 제1 빈은 정규 모드에서 인코딩 및 디코딩되고, 상기 신택스 요소를 표현하기 위해 이용된 하나 이상의 나머지 빈은 바이패스 모드에서 인코딩 및 디코딩된다.

[0222] 일 실시예에서, 대칭 모션 벡터 차이 코딩 모드가 현재 코딩 유닛에 대해 이용되는지를 표시하는 신택스 요소는 바이패스 모드에서 인코딩 및 디코딩된다.

[0223] 실시예는 하나 이상의 프로세서에 의해 실행될 때 하나 이상의 프로세서로 하여금 전술한 실시예들 중 임의의 것에 따른 인코딩 방법 또는 디코딩 방법을 수행하게 하는 명령어들을 포함하는 컴퓨터 프로그램을 제공한다. 본 실시예들 중 하나 이상은 또한 전술한 방법들에 따라 비디오 데이터를 인코딩 또는 디코딩하기 위한 명령어들을 저장한 컴퓨터 판독가능 저장 매체를 제공한다. 하나 이상의 실시예는 또한 전술한 방법들에 따라 생성된 비트스트림을 저장한 컴퓨터 판독가능 저장 매체를 제공한다. 하나 이상의 실시예는 또한 전술한 방법들에 따라 생성된 비트스트림을 송신 또는 수신하기 위한 방법 및 장치를 제공한다.

[0224] 다양한 구현들은 디코딩을 수반한다. "디코딩"은, 본 출원에서 이용되는 바와 같이, 디스플레이에 적합한 최종 출력을 생성하기 위해, 예를 들어, 수신된 인코딩된 시퀀스에 대해 수행된 프로세스들의 전부 또는 일부를 포함할 수 있다. 다양한 실시예들에서, 그러한 프로세스들은 디코더에 의해 전형적으로 수행된 프로세스들, 예를 들어, 엔트로피 디코딩, 역 양자화, 역 변환, 및 차동 디코딩 중 하나 이상을 포함한다. "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.

[0225] 다양한 구현들은 인코딩을 수반한다. "디코딩"에 관한 상기의 논의와 유사한 방식으로, 본 출원에서 이용되는 "인코딩"은 인코딩된 비트스트림을 생성하기 위해, 예를 들어, 입력 비디오 시퀀스에 대해 수행된 프로세스들의 전부 또는 일부를 포함할 수 있다.

[0226] 본 명세서에서 이용된 바와 같은 신택스 요소들, 예를 들어, GBi 인덱스를 특성화하는데 이용된 신택스는 설명적 용어들이라는 점에 유의한다. 그와 같이, 이들은 다른 신택스 요소 명칭들의 이용을 배제하지 않는다.

[0227] 본 명세서에 설명된 구현들 및 양태들은, 예를 들어, 방법 또는 프로세스, 장치, 소프트웨어 프로그램, 데이터 스트림, 또는 신호로 구현될 수 있다. 비록, 단지 단일 형태의 구현의 맥락에서 논의될지라도(예를 들어, 단지 방법으로서 논의될지라도), 논의된 특징들의 구현은 다른 형태들(예를 들어, 장치 또는 프로그램)으로 구현될 수 있다. 장치는, 예를 들어, 적절한 하드웨어, 소프트웨어 및 펌웨어로 구현될 수 있다. 방법들은, 예를 들어, 컴퓨터, 마이크로프로세서, 집적 회로, 또는 프로그래밍가능 로직 디바이스를 포함하는, 일반적으로 처리 디바이스들을 지칭하는 장치, 예를 들어, 프로세서에서 구현될 수 있다. 프로세서들은 또한 통신 디바이스들, 예를 들어, 컴퓨터들, 셀 폰들, 휴대용/개인 휴대 정보 단말기들("PDA들"), 및 최종 사용자들 사이의 정보의 통신을 용이하게 하는 다른 디바이스들을 포함한다.

[0228] "일 실시예" 또는 "실시예" 또는 "일 구현" 또는 "구현" 뿐만 아니라 그의 다른 변형들에 대한 참조는, 실시예와 관련하여 설명된 특정한 특징, 구조, 특성 등이 적어도 하나의 실시예에 포함된다는 것을 의미한다. 따라서, 본 출원 전반에 걸친 다양한 위치들에서 등장하는 구문 "일 실시예에서" 또는 "실시예에서" 또는 "일 구현에서" 또는 "구현에서" 뿐만 아니라 임의의 다른 변형들의 등장은 반드시 모두 동일한 실시예를 참조하는 것은 아니다.

[0229] 또한, 본 출원은 다양한 정보 피스들을 "결정"하는 것을 언급할 수 있다. 정보를 결정하는 것은, 예를 들어, 정보를 추정하는 것, 정보를 계산하는 것, 정보를 예측하는 것, 또는 메모리로부터 정보를 검색하는 것 중 하나 이상을 포함할 수 있다.

[0230] 또한, 본 출원은 다양한 정보 피스들에 "액세스"하는 것을 언급할 수 있다. 정보에 액세스하는 것은, 예를 들어, 정보를 수신하는 것, (예를 들어, 메모리로부터) 정보를 검색하는 것, 정보를 저장하는 것, 정보를 이동시키는 것, 정보를 복사하는 것, 정보를 계산하는 것, 정보를 결정하는 것, 정보를 예측하는 것, 또는 정보를 추정하는 것 중 하나 이상을 포함할 수 있다.

[0231] 또한, 본 출원은 다양한 정보 피스들을 "수신"하는 것을 언급할 수 있다. 수신하는 것은, "액세스하는 것"에서와 같이 광의의 용어로 의도된다. 정보를 수신하는 것은, 예를 들어, 정보을 액세스하는 것 또는 (예를 들어,

메모리로부터) 정보를 검색하는 것 중 하나 이상을 포함할 수 있다. 또한, "수신"하는 것은, 예를 들어, 정보를 저장하는 것, 정보를 처리하는 것, 정보를 송신하는 것, 정보를 이동시키는 것, 정보를 복사하는 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 전형적으로 수반된다.

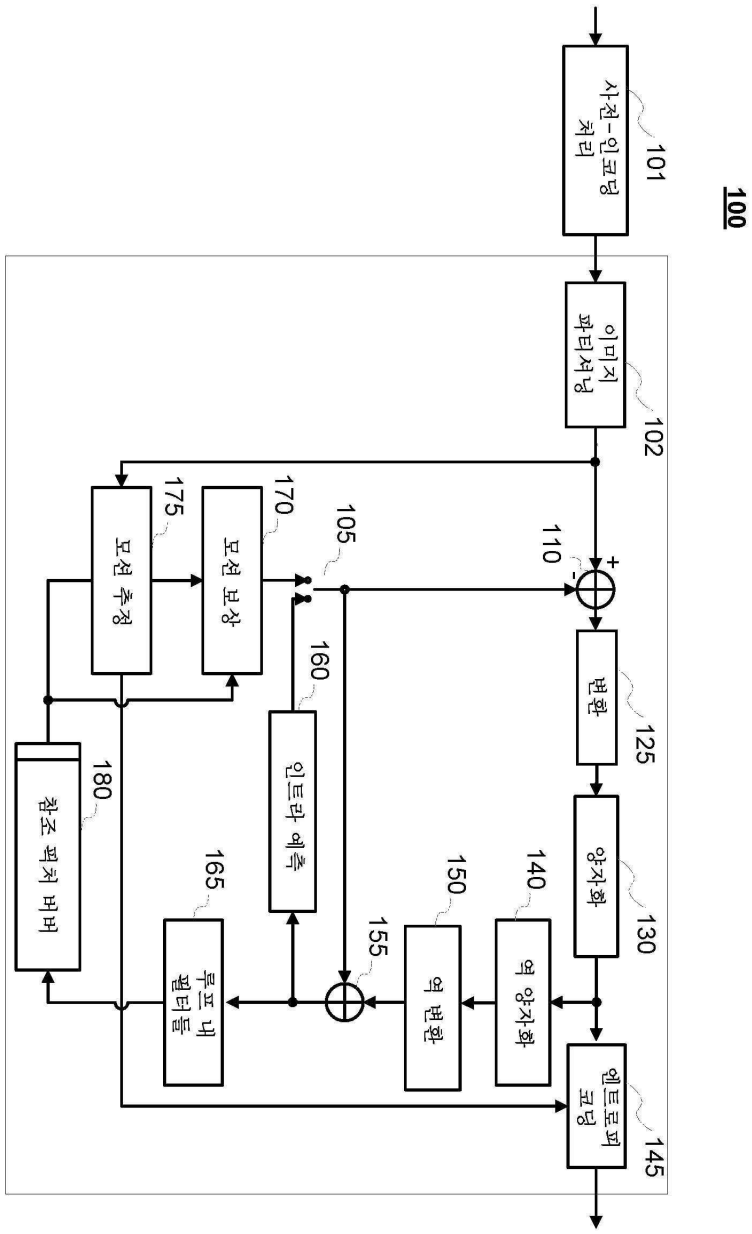
[0232] 예를 들어, "A/B", "A 및/또는 B" 및 "A 및 B 중 적어도 하나"의 경우들에서, 이하의 "/", "및/또는", 및 "~ 중 적어도 하나" 중 임의의 것의 이용은, 첫번째로 열거된 옵션(A)만을 선택함, 또는 두번째로 열거된 옵션(B)만을 선택함, 또는 옵션들 둘다(A 및 B)를 선택함을 포함하려는 의도임을 알 것이다. 추가 예로서, "A, B, 및/또는 C" 및 "A, B, 및 C 중 적어도 하나"의 경우들에서, 이러한 어구는 첫번째로 열거된 옵션(A)만의 선택, 또는 두번째로 열거된 옵션(B)만의 선택, 또는 세번째로 열거된 옵션(C)만의 선택, 또는 첫번째와 두번째로 열거된 옵션들(A 및 B)만의 선택, 또는 첫번째와 세번째로 열거된 옵션들(A 및 C)만의 선택, 또는 두번째와 세번째로 열거된 옵션들(B 및 C)만의 선택, 또는 3개의 옵션들(A 및 B 및 C) 전부의 선택을 포함하는 것으로 의도된다. 이것은 본 기술분야 및 관련 기술분야의 통상의 기술자에게 명백한 바와 같이 열거된 많은 항목들에 대해 확장될 수 있다.

[0233] 또한, 본 명세서에 이용된 바와 같이, 단어 "신호"는 다른 것들 중에서, 대응하는 디코더에게 무언가를 표시하는 것을 지칭한다. 예를 들어, 특정 실시예들에서, 인코더는 구분적 선형 모델에서의 조각들의 수를 디코더에 시그널링한다. 이러한 방식으로, 실시예에서, 인코더 측 및 디코더 측 둘다에서 동일한 파라미터가 이용된다. 따라서, 예를 들어, 인코더는 디코더가 동일한 특정 파라미터를 이용할 수 있도록 디코더에 특정 파라미터를 송신(명시적 시그널링)할 수 있다. 반대로, 디코더가 이미 특정 파라미터 뿐만 아니라 다른 것들을 갖는 경우, 단순히 디코더가 특정 파라미터를 알고 선택하는 것을 허용하기 위해 송신(암시적 시그널링) 없이 시그널링이 이용될 수 있다. 임의의 실제 기능들의 송신을 회피함으로써, 다양한 실시예들에서 비트 절약이 실현된다. 시그널링은 다양한 방식들로 달성될 수 있다는 것을 알아야 한다. 예를 들어, 다양한 실시예들에서, 하나 이상의 신택스 요소, 플래그 등이 정보를 대응하는 디코더에 시그널링하는데 이용된다. 전술한 것은 단어 "신호"의 동사 형태와 관련되지만, 단어 "신호"는 본 명세서에서 명사로서 이용될 수도 있다.

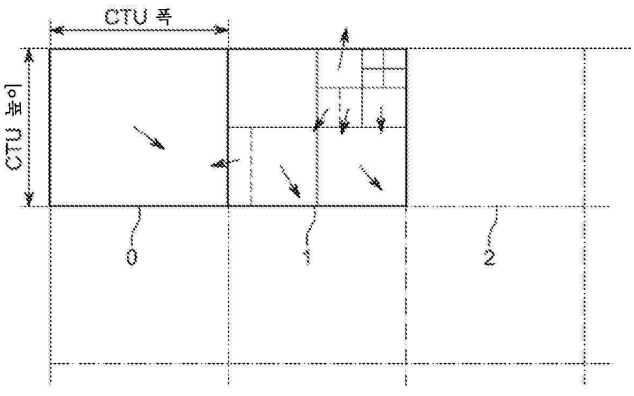
[0234] 본 기술분야의 통상의 기술자에게 명백한 바와 같이, 구현들은, 예를 들어, 저장되거나 송신될 수 있는 정보를 운반하도록 포맷팅된 다양한 신호들을 생성할 수 있다. 정보는, 예를 들어, 방법을 수행하기 위한 명령어들, 또는 설명된 구현들 중 하나에 의해 생성된 데이터를 포함할 수 있다. 예를 들어, 신호는 설명된 실시예의 비트스트림을 운반하도록 포맷팅될 수 있다. 그러한 신호는, 예를 들어, (예컨대, 스펙트럼의 라디오 주파수 부분을 이용하여) 전자기파로서 또는 기저대역 신호로서 포맷팅될 수 있다. 포맷팅은, 예를 들어, 데이터 스트림을 인코딩하고 인코딩된 데이터 스트림으로 캐리어를 변조하는 것을 포함할 수 있다. 신호가 운반하는 정보는, 예를 들어, 아날로그 또는 디지털 정보일 수 있다. 신호는 알려진 바와 같은 다양한 상이한 유선 또는 무선 링크들을 통해 송신될 수 있다. 신호는 프로세서 판독가능 매체상에 저장될 수 있다.

도면

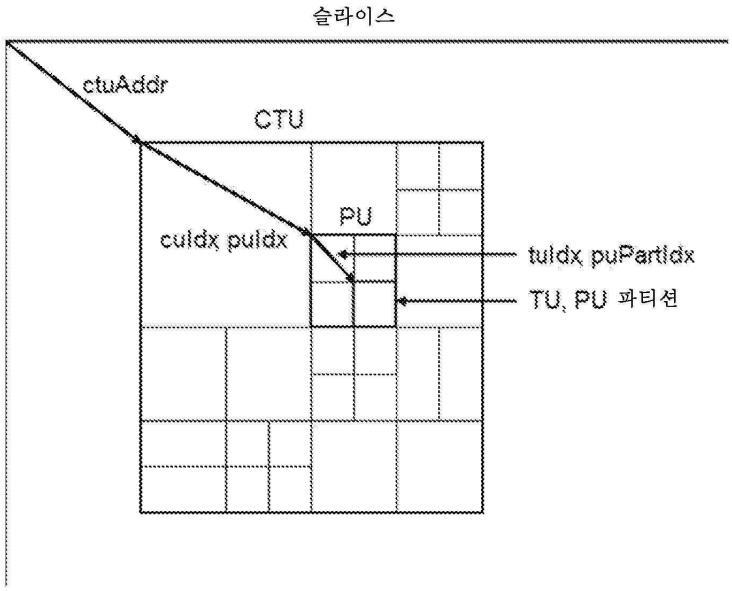
도면1


100
사전-인코딩 처리
101
이미지 파티셔닝
102
모션 추정
175
모션 보상
170
105
110
+
-
변환
125
양자화
130
엔트로피 코딩
145
역 양자화
140
역 변환
150
155
루프 내 필터들
165
인트라 예측
160
참조 픽처 버퍼
180

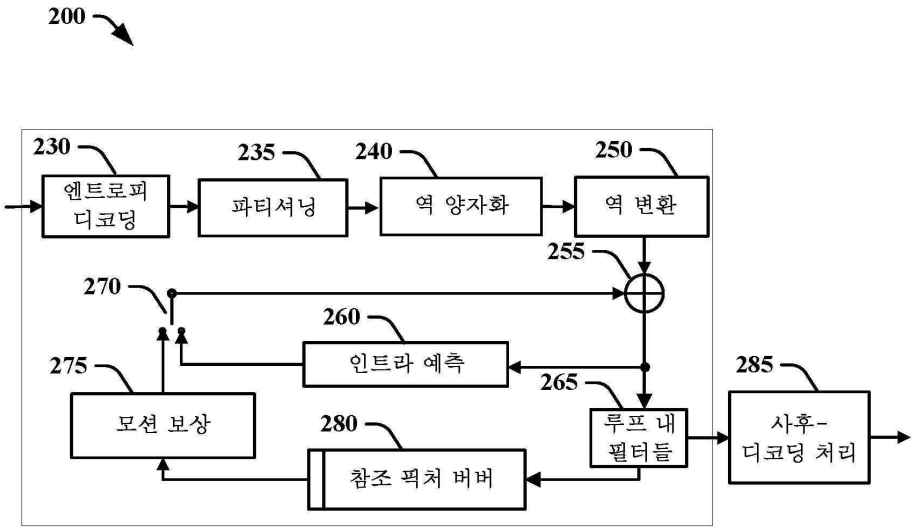
도면2


CTU 폭
CTU 높이

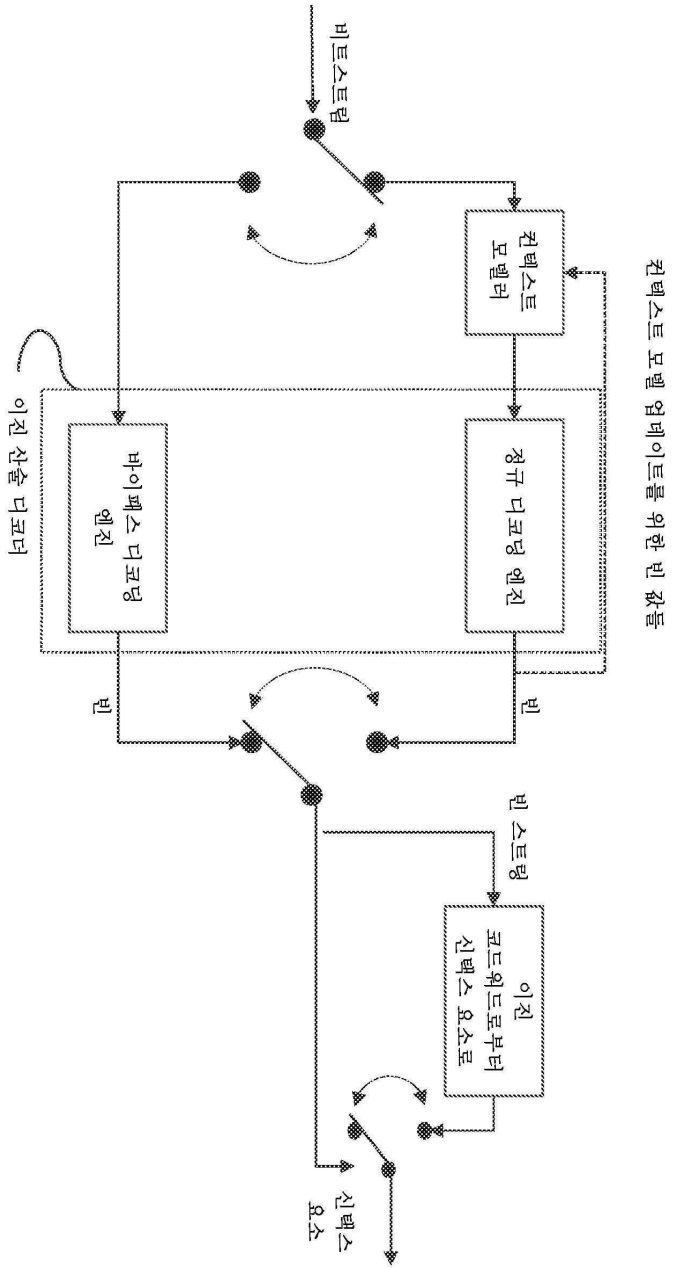
도면3


슬라이스
ctuAddr
CTU
PU
cuIdx puIdx
tuIdx puPartIdx
TU, PU 파티션

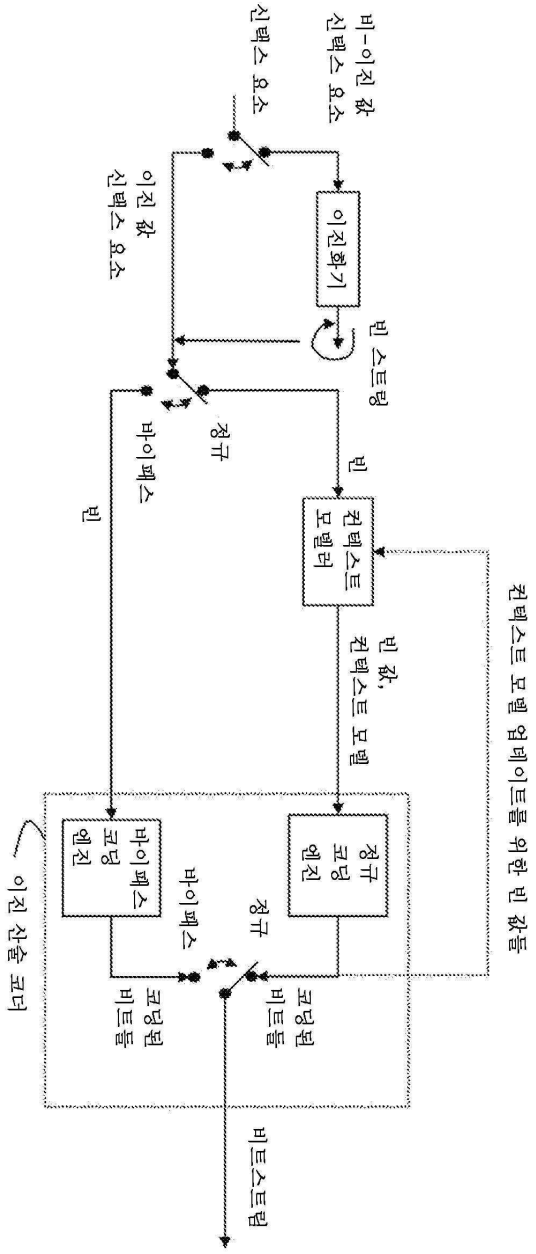
도면4


200
230
엔트로피 디코딩
235
파티셔닝
240
역 양자화
250
역 변환
255
270
260
인트라 예측
275
모션 보상
265
280
참조 픽처 버퍼
루프 내 필터들
285
사후-디코딩 처리

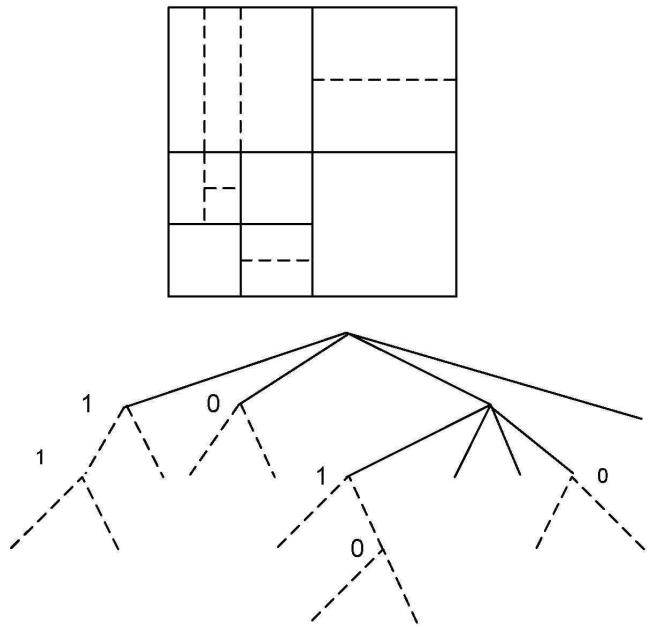
도면5


비트스트림
컨텍스트 모델러
컨텍스트 모델 업데이트를 위한 빈 값들
이진 산술 디코더
바이패스 디코딩 엔진
정규 디코딩 엔진
빈
빈
빈 스트링
이진 코드워드로부터 신택스 요소로
신택스 요소

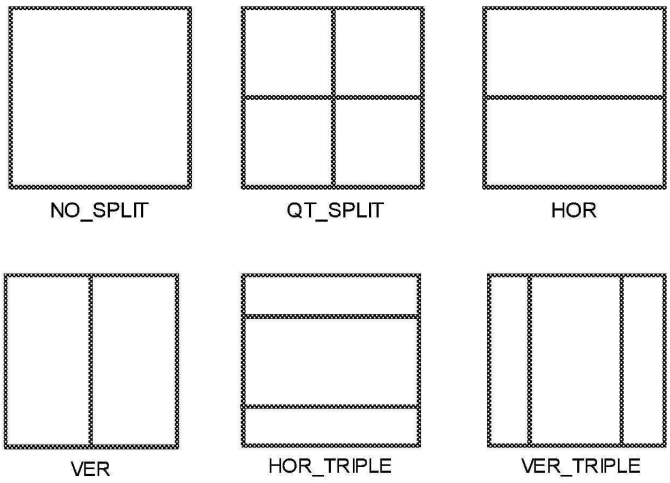
도면6


비-이진 값
신택스 요소
이진 값
신택스 요소
이진화기
빈 스트링
정규
바이패스
빈
빈
컨텍스트
모델러
빈 값,
컨텍스트 모델
컨텍스트 모델 업데이트를 위한 빈 값들
바이패스
코딩
엔진
정규
코딩
엔진
이진 산술 코더
바이패스
정규
코딩된
비트들
코딩된
비트들
비트스트림

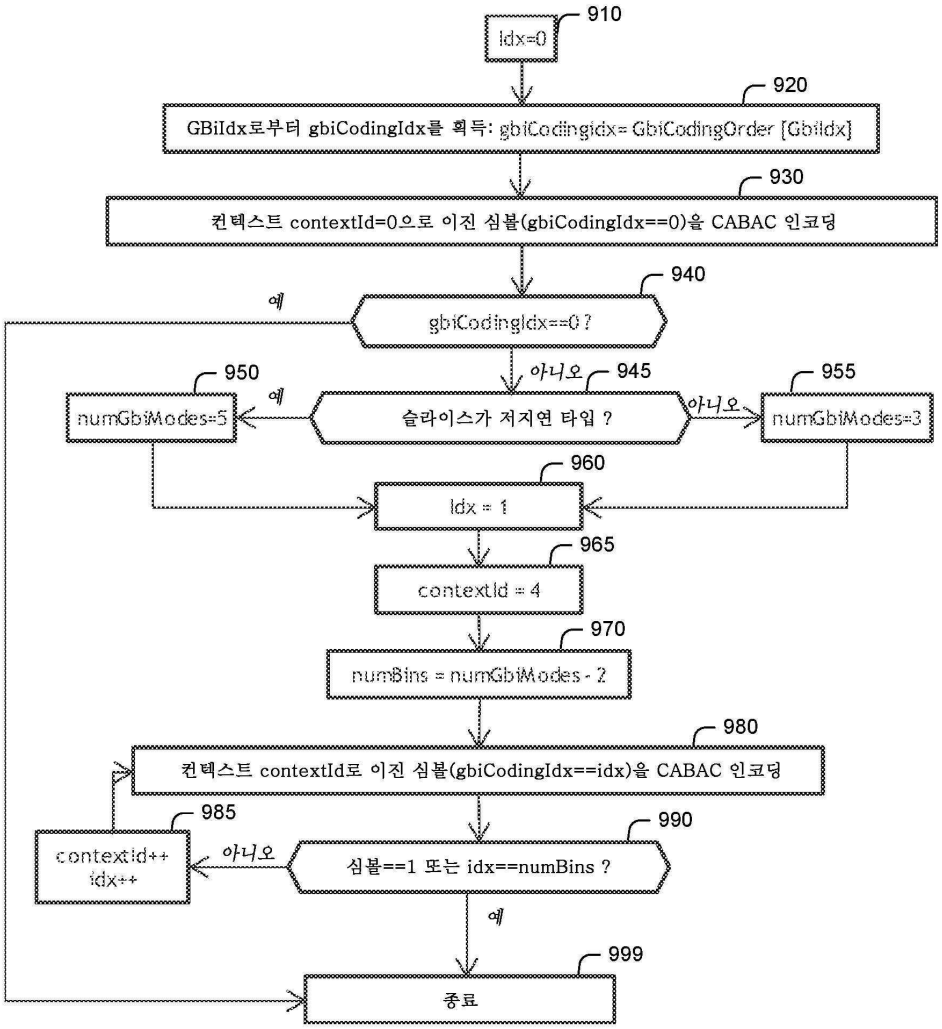
도면7



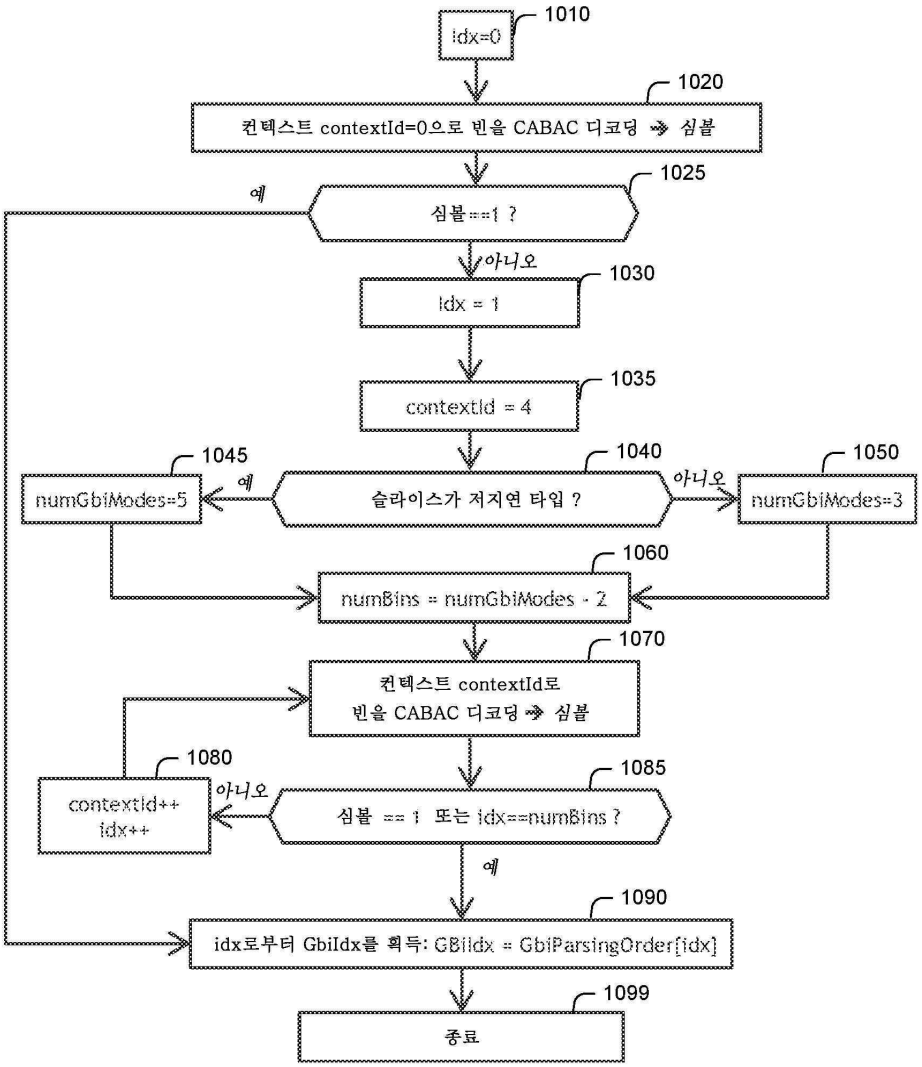
도면8



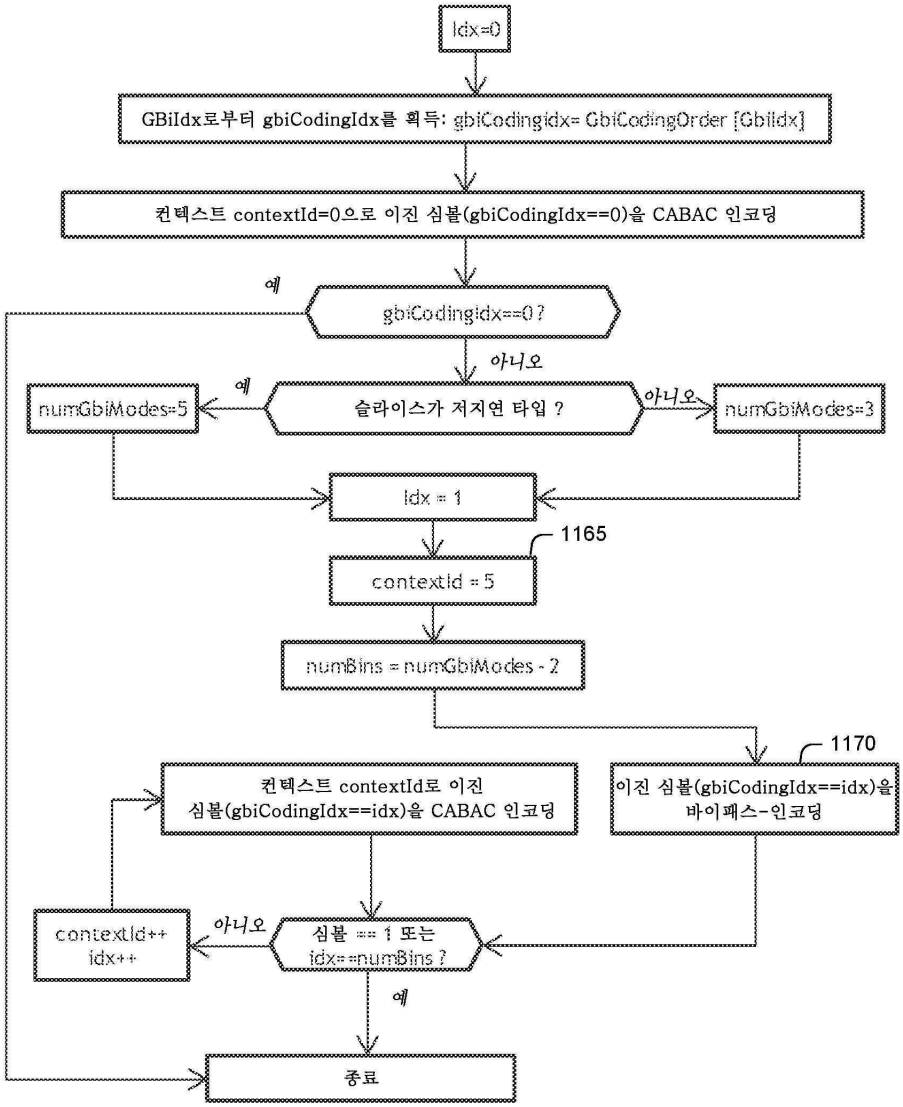
도면9


idx=0
910
GBiIdx로부터 gbiCodingIdx를 획득: gbiCodingIdx= GbiCodingOrder [GbiIdx]
920
컨텍스트 contextId=0으로 이진 심볼(gbiCodingIdx==0)을 CABAC 인코딩
930
gbiCodingIdx==0 ?
940
예
아니오
numGbiModes=5
950
슬라이스가 저지연 타입 ?
945
numGbiModes=3
955
idx = 1
960
contextId = 4
965
numBins = numGbiModes - 2
970
컨텍스트 contextId로 이진 심볼(gbiCodingIdx==idx)을 CABAC 인코딩
980
contextId++
idx++
985
심볼==1 또는 idx==numBins ?
990
종료
999

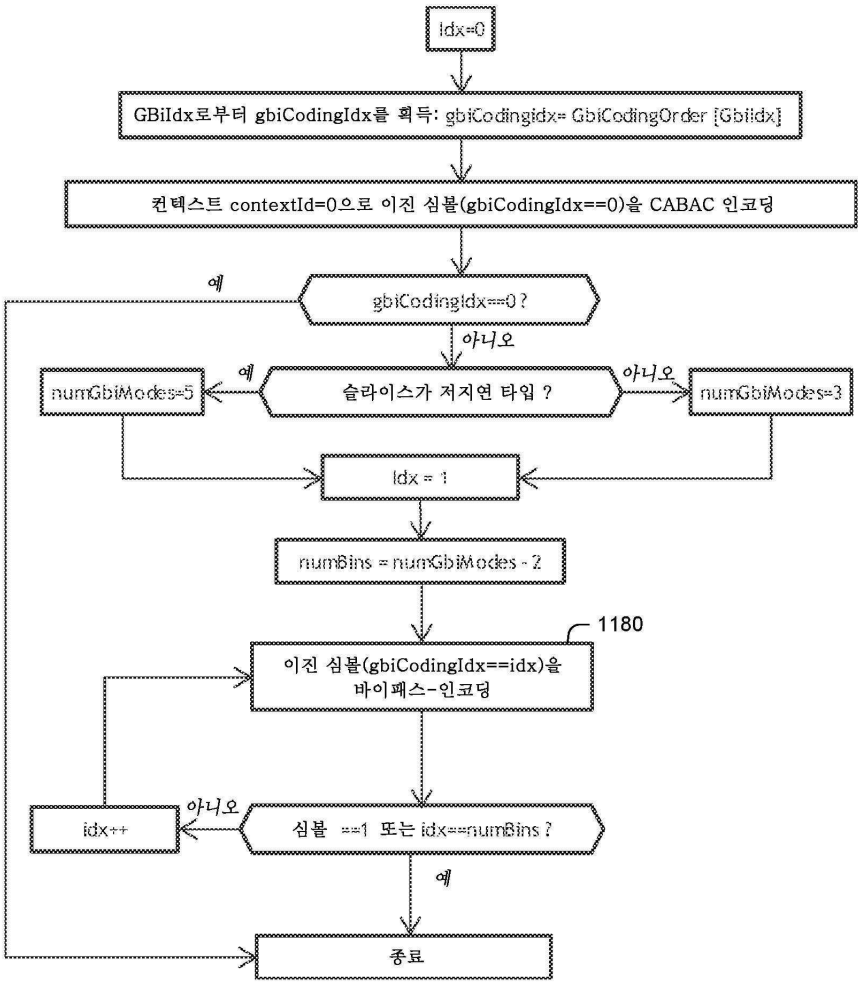
도면10


idx=0
1010
컨텍스트 contextId=0으로 빈을 CABAC 디코딩 ➔ 심볼
1020
심볼==1 ?
1025
예
아니오
idx = 1
1030
contextId = 4
1035
numGbiModes=5
1045
예
슬라이스가 저지연 타입 ?
1040
아니오
numGbiModes=3
1050
numBins = numGbiModes - 2
1060
컨텍스트 contextId로
빈을 CABAC 디코딩 ➔ 심볼
1070
contextId++
idx++
1080
아니오
심볼 == 1 또는 idx==numBins ?
1085
예
idx로부터 GbiIdx를 획득: GBiIdx = GbiParsingOrder[idx]
1090
종료
1099

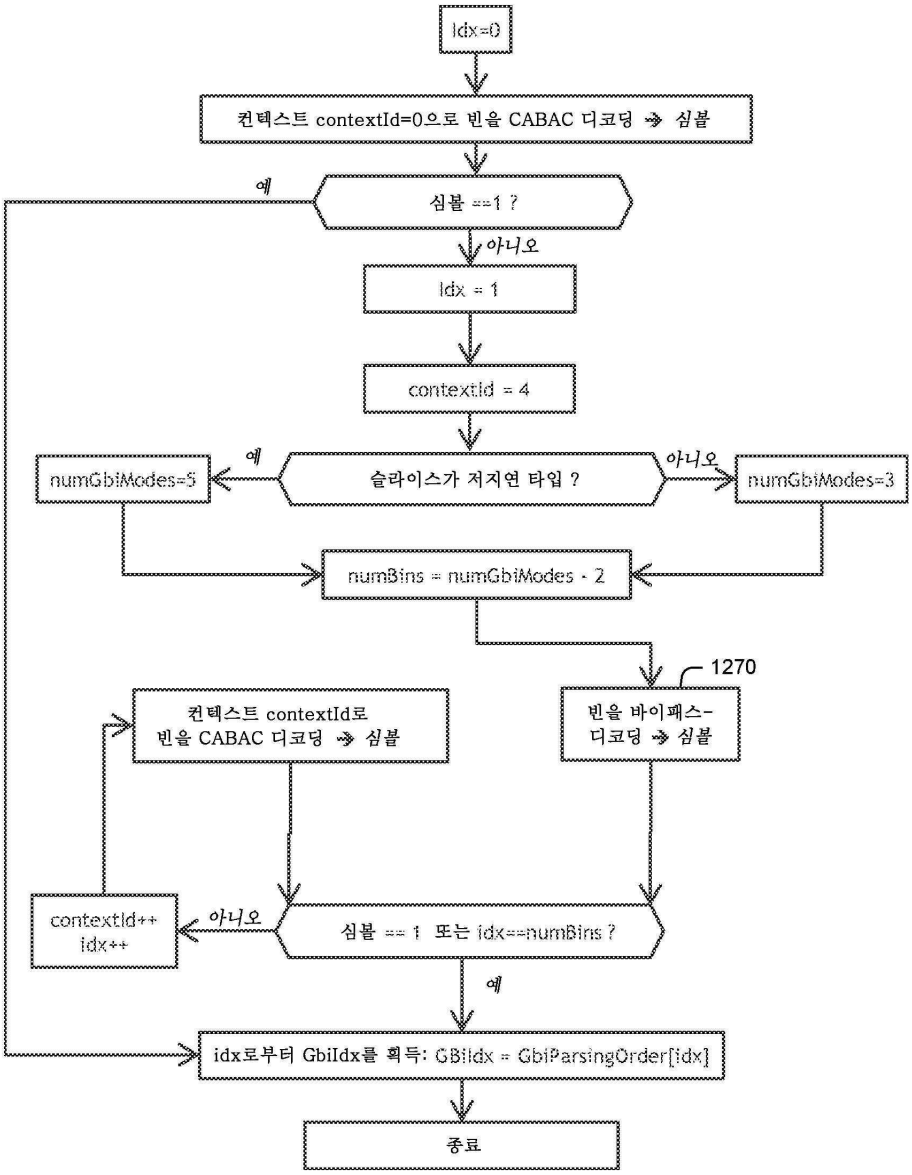
도면11


Idx=0
GBiIdx로부터 gbiCodingIdx를 획득: gbiCodingIdx= GbiCodingOrder [GbiIdx]
컨텍스트 contextId=0으로 이진 심볼(gbiCodingIdx==0)을 CABAC 인코딩
예
gbiCodingIdx==0 ?
아니오
numGbiModes=5
예
슬라이스가 저지연 타입 ?
아니오
numGbiModes=3
Idx = 1
1165
contextId = 5
numBins = numGbiModes - 2
1170
컨텍스트 contextId로 이진 심볼(gbiCodingIdx==idx)을 CABAC 인코딩
이진 심볼(gbiCodingIdx==idx)을 바이패스-인코딩
contextId++
idx++
아니오
심볼 == 1 또는 idx==numBins ?
예
종료

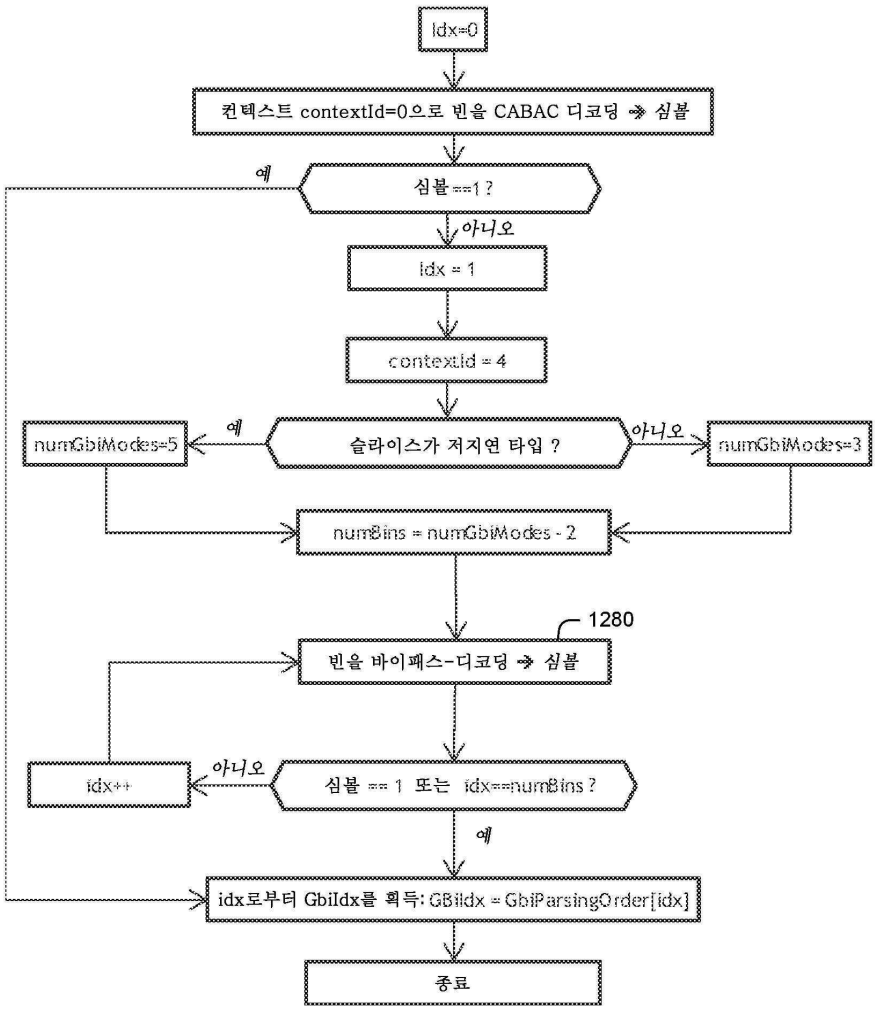
도면11a


Idx=0
GBiIdx로부터 gbiCodingIdx를 획득: gbiCodingIdx= GbiCodingOrder [GbiIdx]
컨텍스트 contextId=0으로 이진 심볼(gbiCodingIdx==0)을 CABAC 인코딩
예
gbiCodingIdx==0 ?
아니오
numGbiModes=5
예
슬라이스가 저지연 타입 ?
아니오
numGbiModes=3
Idx = 1
numBins = numGbiModes - 2
1180
이진 심볼(gbiCodingIdx==idx)을
바이패스-인코딩
idx++
아니오
심볼 ==1 또는 idx==numBins ?
예
종료

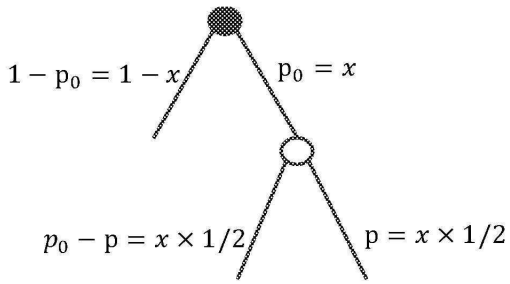
도면12


Idx=0
컨텍스트 contextId=0으로 빈을 CABAC 디코딩 → 심볼
예
심볼 ==1 ?
아니오
Idx = 1
contextId = 4
numGbiModes=5
예
슬라이스가 저지연 타입 ?
아니오
numGbiModes=3
numBins = numGbiModes - 2
1270
컨텍스트 contextId로
빈을 CABAC 디코딩 → 심볼
빈을 바이패스-
디코딩 → 심볼
contextId++
idx++
아니오
심볼 == 1 또는 idx==numBins ?
예
idx로부터 GbiIdx를 획득: GBiIdx = GbiParsingOrder[idx]
종료

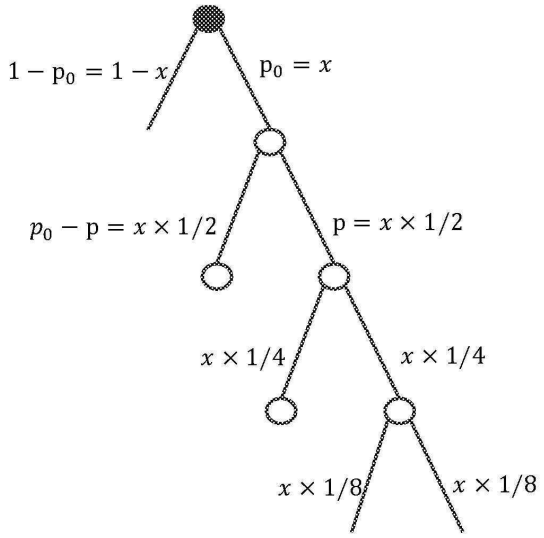
도면12a


idx=0
컨텍스트 contextId=0으로 빈을 CABAC 디코딩 ➔ 심볼
예
심볼==1 ?
아니오
idx = 1
contextId = 4
numGbiModes=5
예
슬라이스가 저지연 타입 ?
아니오
numGbiModes=3
numBins = numGbiModes - 2
1280
빈을 바이패스-디코딩 ➔ 심볼
idx++
아니오
심볼 == 1 또는 idx==numBins ?
예
idx로부터 GbiIdx를 획득: GBiIdx = GbiParsingOrder[idx]
종료

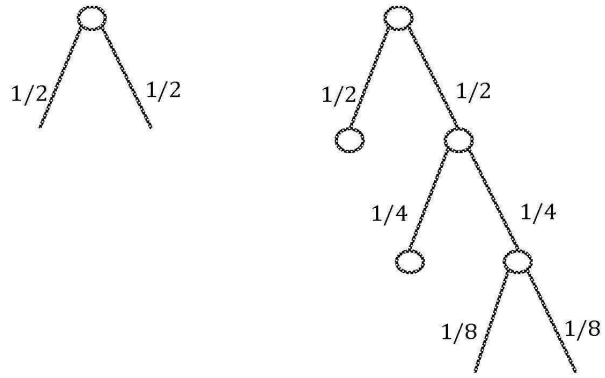
도면12b


$1 - p_0 = 1 - x$
$p_0 = x$
$p_0 - p = x \times 1/2$
$p = x \times 1/2$

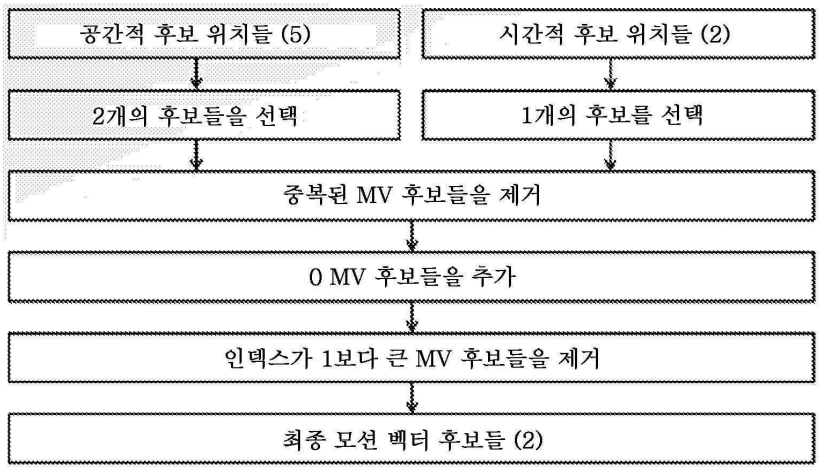
도면12c



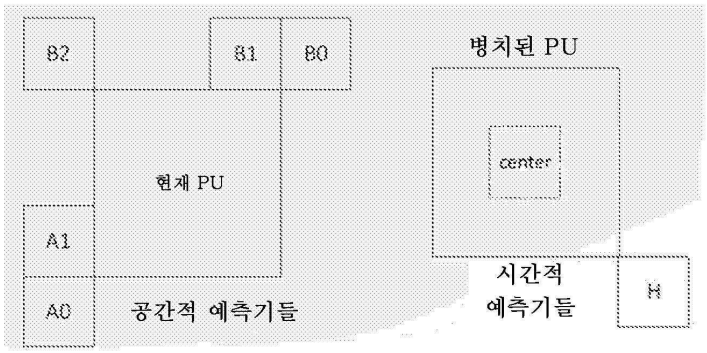
도면12d



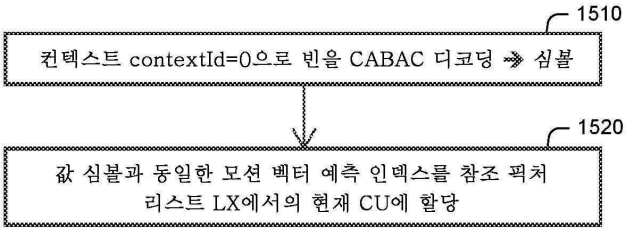
도면13



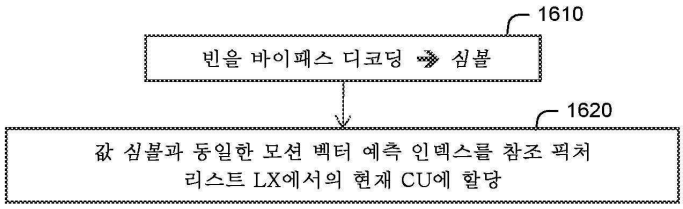
도면14



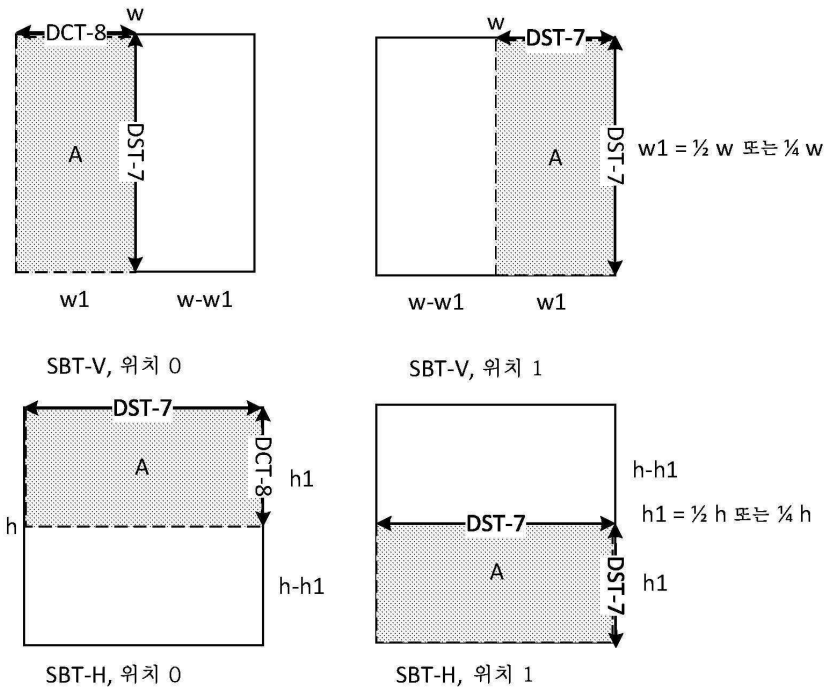
도면15



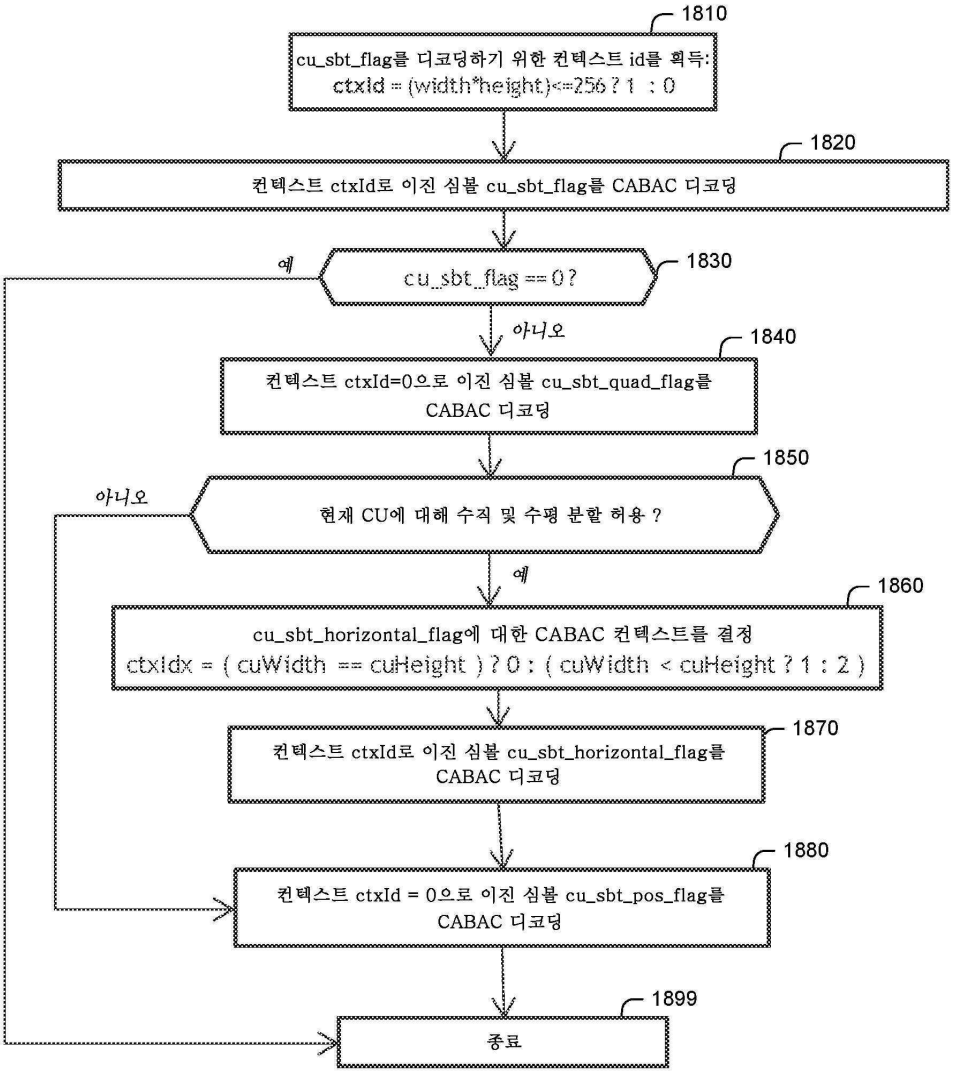
도면16



도면17

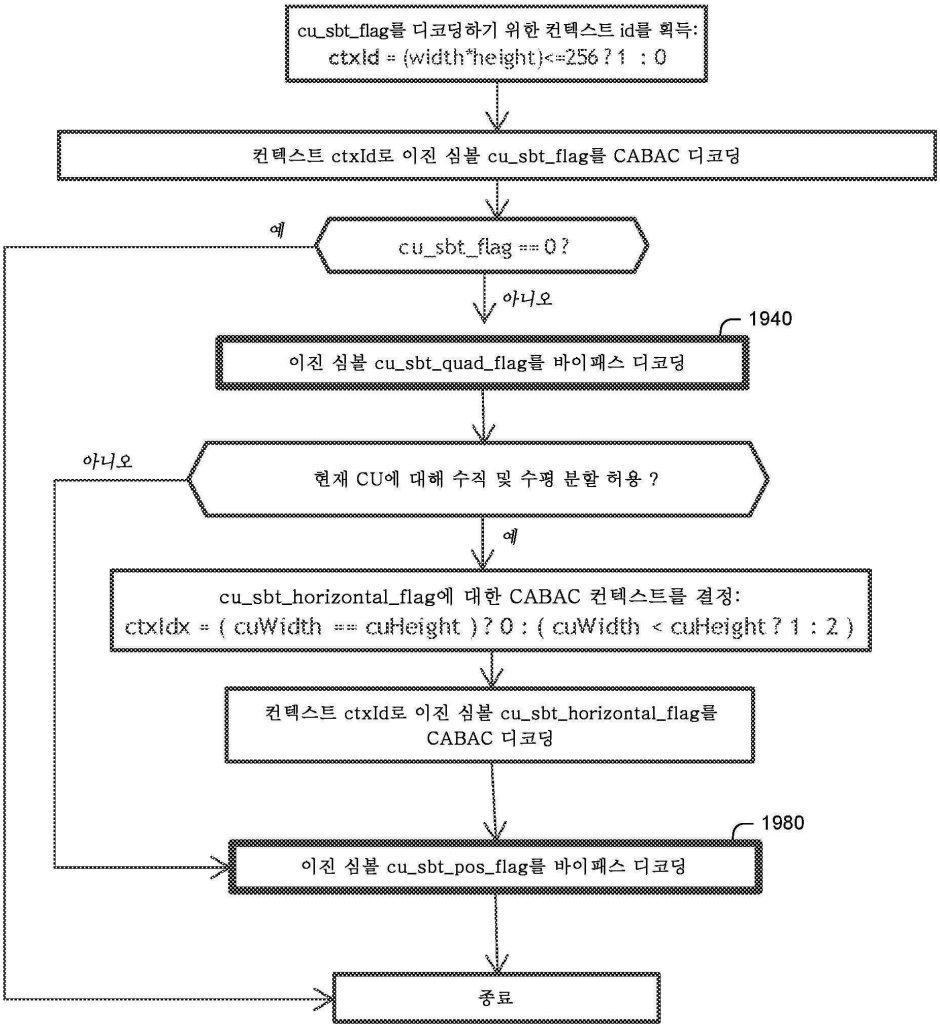

w
DCT-8
A
DST-7
w1
w-w1
SBT-V, 위치 0
w
DST-7
A
DST-7
w1 = ½ w 또는 ¼ w
w-w1
w1
SBT-V, 위치 1
DST-7
A
DCT-8
h1
h
h-h1
SBT-H, 위치 0
h-h1
h1 = ½ h 또는 ¼ h
DST-7
A
DST-7
h1
SBT-H, 위치 1

도면18

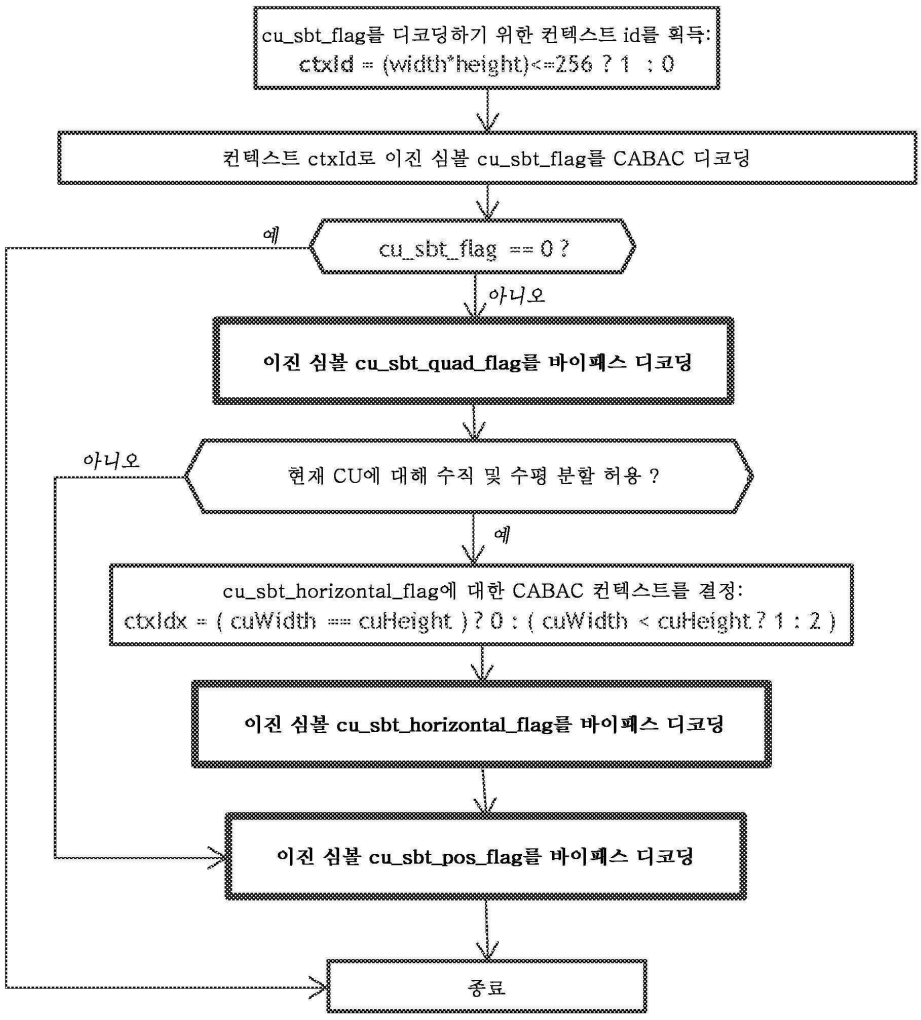

1810
cu_sbt_flag를 디코딩하기 위한 컨텍스트 id를 획득:
ctxId = (width*height)<=256 ? 1 : 0
1820
컨텍스트 ctxId로 이진 심볼 cu_sbt_flag를 CABAC 디코딩
예
cu_sbt_flag == 0?
1830
아니오
1840
컨텍스트 ctxId=0으로 이진 심볼 cu_sbt_quad_flag를
CABAC 디코딩
1850
아니오
현재 CU에 대해 수직 및 수평 분할 허용 ?
예
1860
cu_sbt_horizontal_flag에 대한 CABAC 컨텍스트를 결정
ctxIdx = (cuWidth == cuHeight) ? 0 : (cuWidth < cuHeight ? 1 : 2)
1870
컨텍스트 ctxId로 이진 심볼 cu_sbt_horizontal_flag를
CABAC 디코딩
1880
컨텍스트 ctxId = 0으로 이진 심볼 cu_sbt_pos_flag를
CABAC 디코딩
1899
종료

(종래 기술)

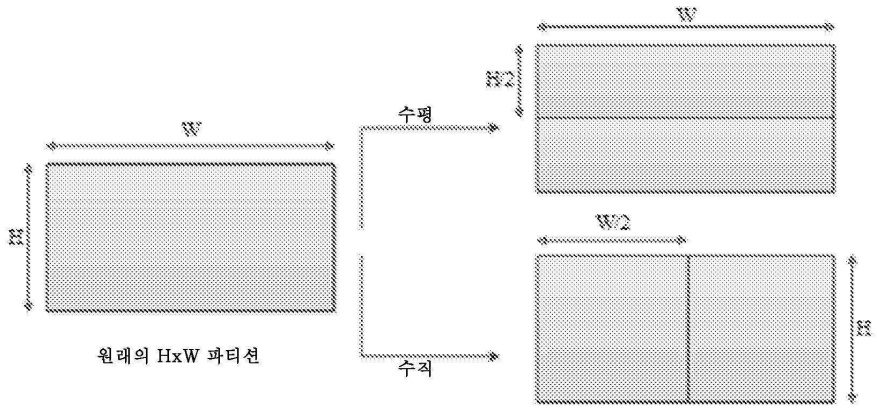
도면19a


cu_sbt_flag를 디코딩하기 위한 컨텍스트 id를 획득:
ctxId = (width*height)<=256 ? 1 : 0
컨텍스트 ctxId로 이진 심볼 cu_sbt_flag를 CABAC 디코딩
예
cu_sbt_flag == 0 ?
아니오
1940
이진 심볼 cu_sbt_quad_flag를 바이패스 디코딩
아니오
현재 CU에 대해 수직 및 수평 분할 허용 ?
예
cu_sbt_horizontal_flag에 대한 CABAC 컨텍스트를 결정:
ctxIdx = (cuWidth == cuHeight) ? 0 : (cuWidth < cuHeight ? 1 : 2)
컨텍스트 ctxId로 이진 심볼 cu_sbt_horizontal_flag를
CABAC 디코딩
1980
이진 심볼 cu_sbt_pos_flag를 바이패스 디코딩
종료

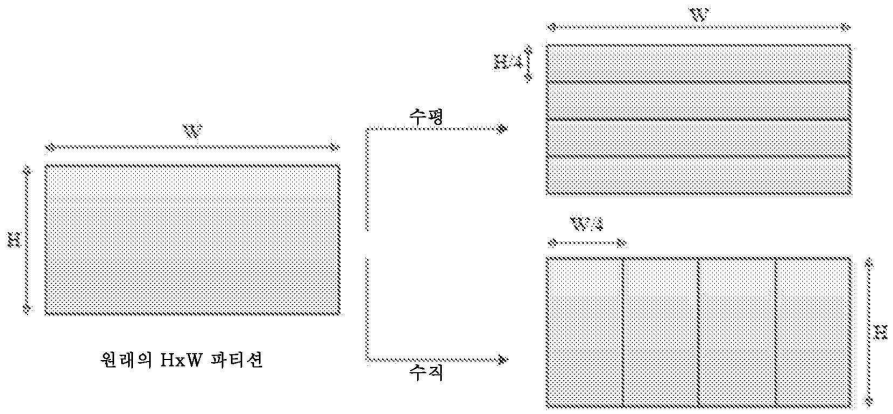
도면19b


cu_sbt_flag를 디코딩하기 위한 컨텍스트 id를 획득:
ctxId = (width*height)<=256 ? 1 : 0
컨텍스트 ctxId로 이진 심볼 cu_sbt_flag를 CABAC 디코딩
예
cu_sbt_flag == 0 ?
아니오
이진 심볼 cu_sbt_quad_flag를 바이패스 디코딩
아니오
현재 CU에 대해 수직 및 수평 분할 허용 ?
예
cu_sbt_horizontal_flag에 대한 CABAC 컨텍스트를 결정:
ctxIdx = (cuWidth == cuHeight) ? 0 : (cuWidth < cuHeight ? 1 : 2)
이진 심볼 cu_sbt_horizontal_flag를 바이패스 디코딩
이진 심볼 cu_sbt_pos_flag를 바이패스 디코딩
종료

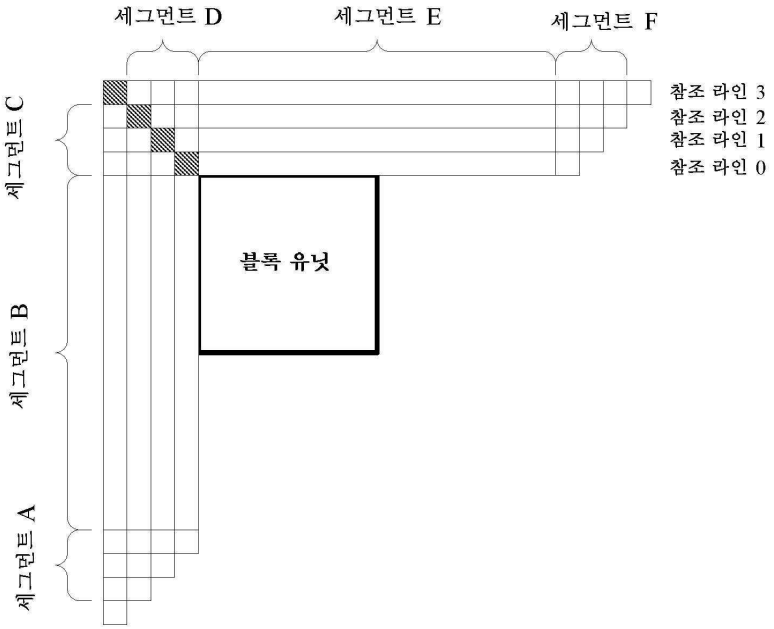
도면20a


W
H
원래의 HxW 파티션
수평
수직
W
H/2
W/2
H

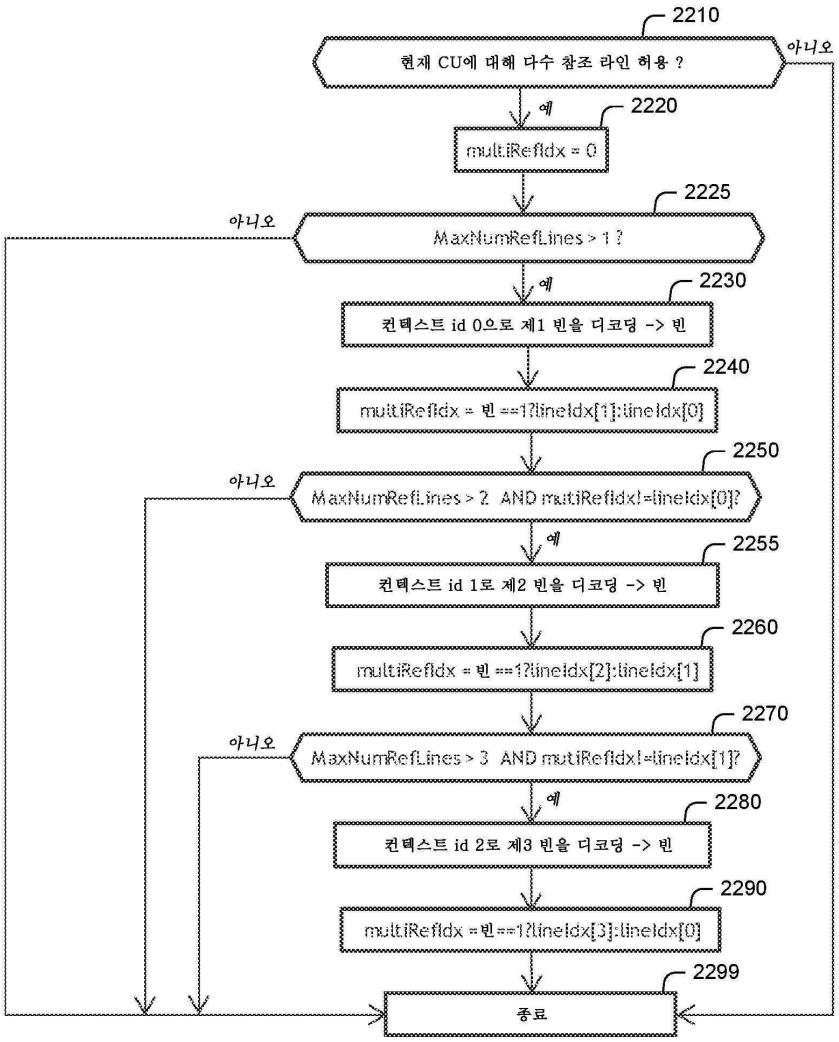
도면20b


W
H/4
수평
W
H
W/4
H
원래의 HxW 파티션
수직

도면21

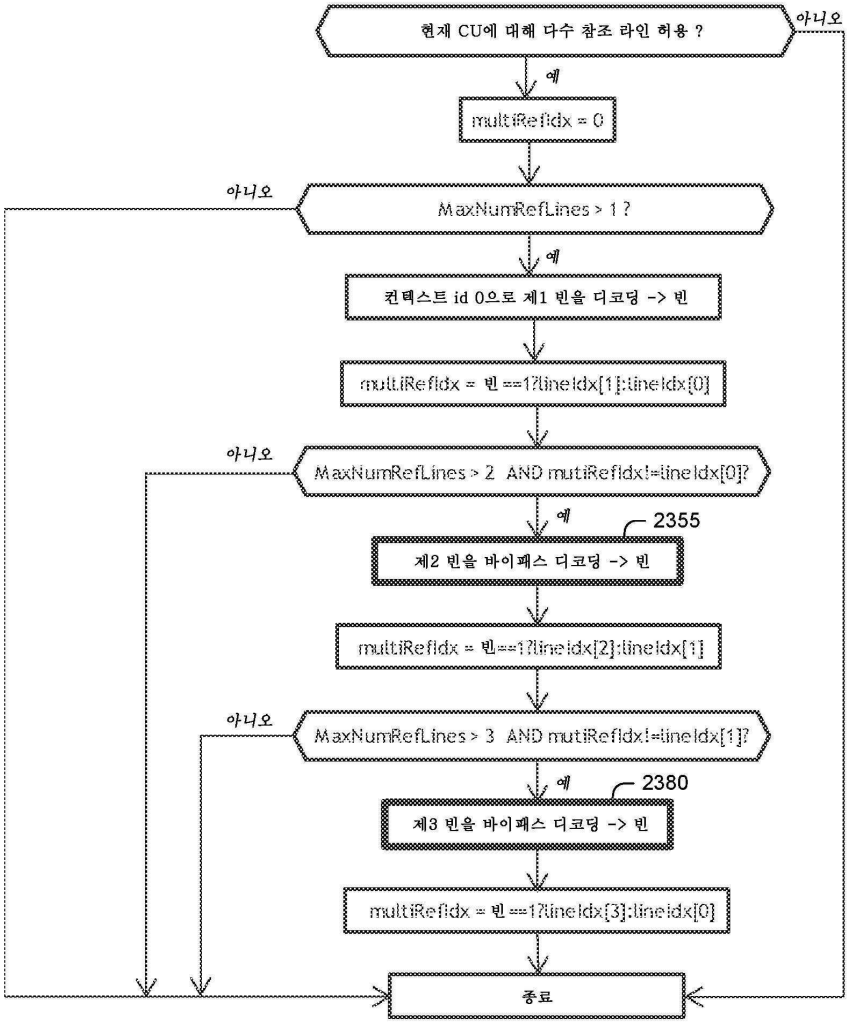

세그먼트 D
세그먼트 E
세그먼트 F
참조 라인 3
참조 라인 2
참조 라인 1
참조 라인 0
세그먼트 C
세그먼트 B
세그먼트 A
블록 유닛

도면22

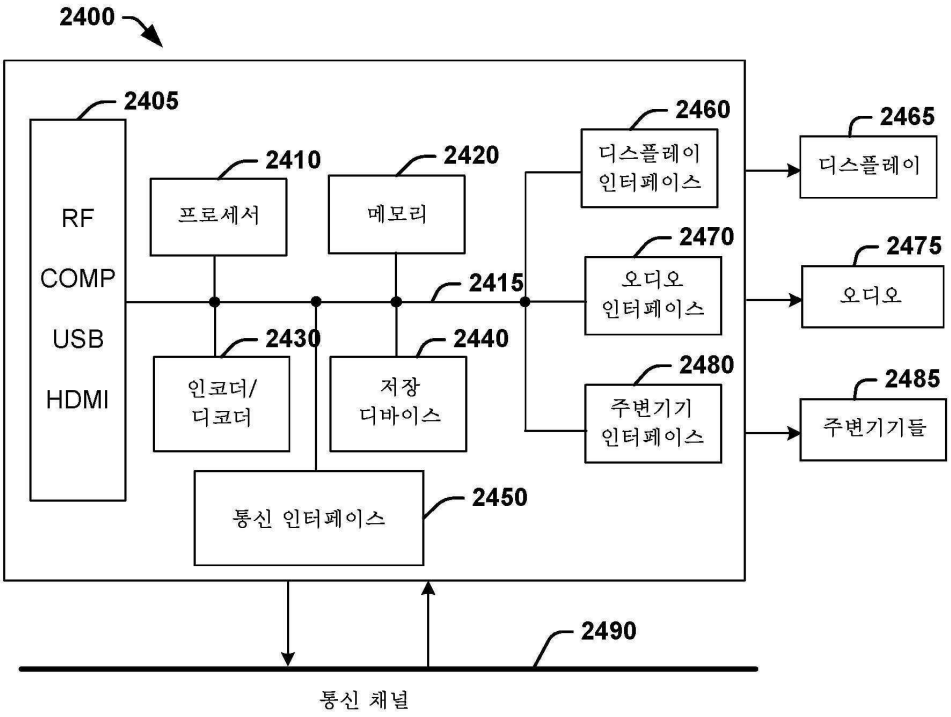

2210
현재 CU에 대해 다수 참조 라인 허용 ?
아니오
예
2220
multiRefIdx = 0
2225
아니오
MaxNumRefLines > 1 ?
예
2230
컨텍스트 id 0으로 제1 빈을 디코딩 -> 빈
2240
multiRefIdx = 빈 ==1?lineIdx[1]:lineIdx[0]
2250
아니오
MaxNumRefLines > 2 AND multiRefIdx!=lineIdx[0]?
예
2255
컨텍스트 id 1로 제2 빈을 디코딩 -> 빈
2260
multiRefIdx = 빈 ==1?lineIdx[2]:lineIdx[1]
2270
아니오
MaxNumRefLines > 3 AND multiRefIdx!=lineIdx[1]?
예
2280
컨텍스트 id 2로 제3 빈을 디코딩 -> 빈
2290
multiRefIdx = 빈==1?lineIdx[3]:lineIdx[0]
2299
종료

(종래 기술)

도면23


현재 CU에 대해 다수 참조 라인 허용 ?
아니오
예
multiRefIdx = 0
MaxNumRefLines > 1 ?
아니오
예
컨텍스트 id 0으로 제1 빈을 디코딩 -> 빈
multiRefIdx = 빈 ==1?lineIdx[1]:lineIdx[0]
MaxNumRefLines > 2 AND mutiRefIdx!=lineIdx[0]?
아니오
예
2355
제2 빈을 바이패스 디코딩 -> 빈
multiRefIdx = 빈==1?lineIdx[2]:lineIdx[1]
MaxNumRefLines > 3 AND mutiRefIdx!=lineIdx[1]?
아니오
예
2380
제3 빈을 바이패스 디코딩 -> 빈
multiRefIdx = 빈 ==1?lineIdx[3]:lineIdx[0]
종료

도면24


2400
2405
RF
COMP
USB
HDMI
2410
프로세서
2420
메모리
2415
2430
인코더/
디코더
2440
저장
디바이스
2450
통신 인터페이스
2460
디스플레이
인터페이스
2470
오디오
인터페이스
2480
주변기기
인터페이스
2465
디스플레이
2475
오디오
2485
주변기기들
2490
통신 채널