



(19) 대한민국특허청(KR)  
(12) 등록특허공보(B1)

(45) 공고일자 2024년07월24일  
(11) 등록번호 10-2688366  
(24) 등록일자 2024년07월22일

(51) 국제특허분류(Int. Cl.)

H04N 19/433 (2014.01) H04N 19/105 (2014.01)  
H04N 19/132 (2014.01) H04N 19/176 (2014.01)  
H04N 19/52 (2014.01) H04N 19/593 (2014.01)  
H04N 19/96 (2014.01)

(52) CPC특허분류

H04N 19/433 (2015.01)  
H04N 19/105 (2015.01)

(21) 출원번호 10-2021-7026799

(22) 출원일자(국제) 2020년03월02일

심사청구일자 2023년02월08일

(85) 번역문제출일자 2021년08월23일

(65) 공개번호 10-2021-0132658

(43) 공개일자 2021년11월04일

(86) 국제출원번호 PCT/CN2020/077415

(87) 국제공개번호 WO 2020/177659

국제공개일자 2020년09월10일

(30) 우선권주장

PCT/CN2019/076695 2019년03월01일 중국(CN)

(56) 선행기술조사문헌

US20150195559 A1

(뒷면에 계속)

전체 청구항 수 : 총 12 항

(73) 특허권자

베이징 바이트댄스 네트워크 테크놀로지 컴퍼니, 리미티드

중국, 베이징 100041, 스징산 디스트릭트, 스징로드, 넘버 30, 넘버 3빌딩, 2층, 룸 비-0035

바이트댄스 아이엔씨

미국, 90066 캘리포니아 로스엔젤레스 스위트룸 137호 6층 제퍼슨 블러바드 웨스트 12655

(72) 발명자

수 지장

미국 90066 캘리포니아 로스엔젤레스 스위트룸 137호 6층 제퍼슨 블러바드 웨스트 12655

장 리

미국 90066 캘리포니아 로스엔젤레스 스위트룸 137호 6층 제퍼슨 블러바드 웨스트 12655

(뒷면에 계속)

(74) 대리인

성병기

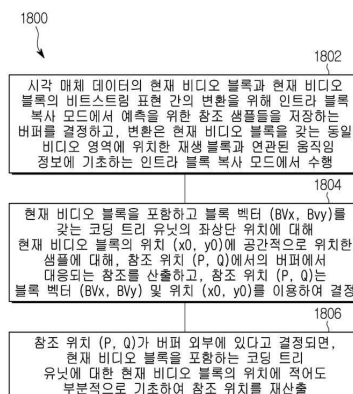
심사관 : 이남숙

(54) 발명의 명칭 비디오 코딩에서 인트라 블록 복사를 위한 방향 기반 예측

(57) 요약

시각 매체 처리 방법은 시각 매체 데이터의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드에서 예측을 위한 참조 샘플들을 저장하는 버퍼를 결정하는 단계; 블록 벡터를 갖고 현재 비디오 블록을 포함하는 코딩 트리 유닛의 좌상단 위치에 대해 현재 비디오 블록의 위치에 공간적으로 위치하는 샘플에 대해, 참조 위치에서 버퍼에서 대응되는 참조를 산출하는 단계, 여기서 참조 위치는 블록 벡터와 위치를 이용하여 결정되고, 참조 위치가 버퍼 외부에 있다고 결정되면, 현재 비디오 블록을 포함하는 코딩 트리 유닛에 대한 현재 비디오 블록의 위치에 적어도 부분적으로 기초하여 참조 위치를 재산출하는 단계를 포함한다.

대표도 - 도18



(52) CPC특허분류

*H04N 19/132* (2015.01)

*H04N 19/176* (2015.01)

*H04N 19/52* (2015.01)

*H04N 19/593* (2015.01)

*H04N 19/96* (2015.01)

(72) 발명자

**장 카이**

미국 90066 캘리포니아 로스엔젤레스 스위트룸 13  
7호 6층 제퍼슨 블러바드 웨스트 12655

**리우 홍빈**

진르터우타오 포스트 오피스 차이나 세틀라이트 커  
뮤니케이션 타워 63 지춘 로드 하이디안 디스트릭  
트 베이징 100080 중국

**왕 유에**

진르터우타오 포스트 오피스 차이나 세틀라이트 커  
뮤니케이션 타워 63 지춘 로드 하이디안 디스트릭  
트 베이징 100080 중국

(56) 선행기술조사문헌

US20160337661 A1\*

US20170223379 A1\*

Xiaozhong Xu, et al., CE8: CPR reference  
memory reuse without increasing memory  
requirement (CE8.1.2a and CE8.1.2d),  
JVET-M0407-v3, 2019.01.17.\*

Benjamin Bross, et al., Versatile Video Coding  
(Draft 4), JVET-M1001-v5, 2019.02.27.\*

\*는 심사관에 의하여 인용된 문헌

## 명세서

### 청구범위

#### 청구항 1

시각 매체 처리 방법에 있어서,

비디오의 현재 비디오 블록과 상기 비디오의 비트스트림의 변환을 위해, 제1 코딩 모드가 상기 현재 비디오 블록에 적용됨을 결정하는 단계;

상기 현재 비디오 블록에 대한 제1 블록 벡터 (BVx, BVy) 를 유도하는 단계;

상기 제1 블록 벡터와 샘플 버퍼에 기초하여 상기 현재 비디오 블록에 대해 예측 샘플들을 생성하는 단계, 필터링 연산이 적용되지 않은 이전 비디오 블록들의 복원 샘플들은 상기 샘플 버퍼에 저장되고, 상기 제1 코딩 모드에서 상기 예측 샘플들은 상기 현재 비디오 블록을 포함하는 동일 픽처로부터 유도되며, 상기 샘플 버퍼의 비트 깊이는 상기 변환 동안 사용된 복원 버퍼의 비트 깊이와 동일하고; 및

상기 예측 샘플들에 기초하여 상기 변환을 수행하는 단계;를 포함하되,

상기 현재 비디오 블록에서 제1 샘플 (x0, y0)의 제1 예측 샘플을 생성하기 위해, 상기 제1 예측 샘플의 상기 샘플 버퍼에서의 위치를 유도하기 위해 위치 변환 연산이 (x0+BVx, y0+BVy)에 적용되고,

상기 위치 변환 연산은 모듈로 연산이고, 상기 제1 예측 샘플의 상기 위치는 ((x0+BVx) mod M, (y0+BVy) mod N)로 표현되며, mod는 모듈로 함수이며,

M 및 N은 상기 현재 비디오 블록을 포함하는 코딩 트리 블록의 크기와 관련된, 시각 매체 처리 방법.

#### 청구항 2

제1 항에 있어서,

상기 샘플 버퍼는 제1 순서로 업데이트되는, 시각 매체 처리 방법.

#### 청구항 3

제2 항에 있어서,

상기 제1 순서는 상기 변환 동안 샘플 복원 순서인, 시각 매체 처리 방법.

#### 청구항 4

제2 항에 있어서,

상기 제1 순서는 선입선출(first-in-first-out: FIFO) 규칙에 기초하는, 시각 매체 처리 방법.

#### 청구항 5

제4 항에 있어서,

상기 샘플 버퍼가 가득차는 경우, 상기 샘플 버퍼에 가장 먼저 추가된 상기 샘플들은 가장 최근의 복원 샘플들로 대체되는, 시각 매체 처리 방법.

## 청구항 6

제1 항에 있어서,

상기 샘플 버퍼는 직사각형 영역이고, 상기 샘플 버퍼의 크기는 상기 비트스트림에 포함된 필드에 기초하여 지시되는, 시각 매체 처리 방법.

## 청구항 7

제1 항에 있어서,

상기 이전 비디오 블록들의 일부들은 상기 현재 비디오 블록을 포함하는 현재 코딩 트리 블록과 상이한 코딩 트리 블록에 위치하는, 시각 매체 처리 방법.

## 청구항 8

제1 항에 있어서,

상기 변환은 상기 현재 비디오 블록을 상기 비트스트림으로 인코딩하는 것을 포함하는, 시각 매체 처리 방법.

## 청구항 9

제1 항에 있어서,

상기 변환은 상기 비트스트림으로부터 상기 현재 비디오 블록을 디코딩하는 것을 포함하는, 시각 매체 처리 방법.

## 청구항 10

프로세서 및 명령이 포함된 비 일시적 메모리를 포함하는 비디오 데이터 처리 장치에 있어서, 상기 명령은 상기 프로세서에 의해 실행될 때 상기 프로세서가,

비디오의 현재 비디오 블록과 상기 비디오의 비트스트림의 변환을 위해, 제1 코딩 모드가 상기 현재 비디오 블록에 적용됨을 결정하게 하고;

상기 현재 비디오 블록에 대한 제1 블록 벡터 (BVx, BVy) 를 유도하게 하고;

상기 제1 블록 벡터와 샘플 버퍼에 기초하여 상기 현재 비디오 블록에 대해 예측 샘플들을 생성하게 하되, 필터링 연산이 적용되지 않은 이전 비디오 블록들의 복원 샘플들은 상기 샘플 버퍼에 저장되고, 상기 제1 코딩 모드에서 상기 예측 샘플들은 상기 현재 비디오 블록을 포함하는 동일 픽처로부터 유도되며, 상기 샘플 버퍼의 비트 깊이는 상기 변환 동안 사용된 복원 버퍼의 비트 깊이와 동일하고; 및

상기 예측 샘플들에 기초하여 상기 변환을 수행하되,

상기 현재 비디오 블록에서 제1 샘플 (x0, y0)의 제1 예측 샘플을 생성하기 위해, 상기 제1 예측 샘플의 상기 샘플 버퍼에서의 위치를 유도하기 위해 위치 변환 연산이  $(x0+Bvx, y0+Bvy)$ 에 적용되고,

상기 위치 변환 연산은 모듈로 연산이고, 상기 제1 예측 샘플의 상기 위치는  $((x0+Bvx) \bmod M, (y0+Bvy) \bmod N)$ 로 표현되며, mod는 모듈로 함수이며,

M 및 N은 상기 현재 비디오 블록을 포함하는 코딩 트리 블록의 크기와 관련된, 비디오 데이터 처리 장치.

## 청구항 11

명령을 포함하는 비-일시적 컴퓨터 판독 가능한 저장 매체에 있어서, 상기 명령은 프로세서가,

비디오의 현재 비디오 블록과 상기 비디오의 비트스트림의 변환을 위해, 제1 코딩 모드가 상기 현재 비디오 블록에 적용됨을 결정하게 하고;

상기 현재 비디오 블록에 대한 제1 블록 벡터 (BVx, BVy) 를 유도하게 하고;

상기 제1 블록 벡터와 샘플 버퍼에 기초하여 상기 현재 비디오 블록에 대해 예측 샘플들을 생성하게 하되, 필터링 연산이 적용되지 않은 이전 비디오 블록들의 복원 샘플들은 상기 샘플 버퍼에 저장되고, 상기 제1 코딩 모드에서 상기 예측 샘플들은 상기 현재 비디오 블록을 포함하는 동일 픽처로부터 유도되며, 상기 샘플 버퍼의 비트 깊이는 상기 변환 동안 사용된 복원 버퍼의 비트 깊이와 동일하고; 및

상기 예측 샘플들에 기초하여 상기 변환을 수행하되,

상기 현재 비디오 블록에서 제1 샘플 (x0, y0)의 제1 예측 샘플을 생성하기 위해, 상기 제1 예측 샘플의 상기 샘플 버퍼에서의 위치를 유도하기 위해 위치 변환 연산이 (x0+BVx, y0+BVy)에 적용되고,

상기 위치 변환 연산은 모듈로 연산이고, 상기 제1 예측 샘플의 상기 위치는 ((x0+BVx) mod M, (y0+BVy) mod N)로 표현되며, mod는 모듈로 함수이며,

M 및 N은 상기 현재 비디오 블록을 포함하는 코딩 트리 블록의 크기와 관련된, 비-일시적 컴퓨터 판독 가능한 저장 매체.

## 청구항 12

비디오 처리 장치에 의해 수행되는 방법에 의해 생성된 비디오의 비트스트림을 저장하는 비-일시적 컴퓨터 판독 가능한 기록 매체에 있어서, 상기 방법은,

상기 비디오의 현재 비디오 블록을 위해, 제1 코딩 모드가 상기 현재 비디오 블록에 적용됨을 결정하는 단계;

상기 현재 비디오 블록에 대한 제1 블록 벡터 (BVx, BVy) 를 유도하는 단계;

상기 제1 블록 벡터와 샘플 버퍼에 기초하여 상기 현재 비디오 블록에 대해 예측 샘플들을 생성하는 단계, 필터링 연산이 적용되지 않은 이전 비디오 블록들의 복원 샘플들은 상기 샘플 버퍼에 저장되고, 상기 제1 코딩 모드에서 상기 예측 샘플들은 상기 현재 비디오 블록을 포함하는 동일 픽처로부터 유도되며, 상기 샘플 버퍼의 비트 깊이는 복원 버퍼의 비트 깊이와 동일하고; 및

상기 예측 샘플들에 기초하여 상기 비트스트림을 생성하는 단계;를 포함하되,

상기 현재 비디오 블록에서 제1 샘플 (x0, y0)의 제1 예측 샘플을 생성하기 위해, 상기 제1 예측 샘플의 상기 샘플 버퍼에서의 위치를 유도하기 위해 위치 변환 연산이 (x0+BVx, y0+BVy)에 적용되고,

상기 위치 변환 연산은 모듈로 연산이고, 상기 제1 예측 샘플의 상기 위치는 ((x0+BVx) mod M, (y0+BVy) mod N)로 표현되며, mod는 모듈로 함수이며,

M 및 N은 상기 현재 비디오 블록을 포함하는 코딩 트리 블록의 크기와 관련된, 비-일시적 컴퓨터 판독 가능한 기록 매체.

## 청구항 13

삭제

## 청구항 14

삭제

## 청구항 15

삭제

## 청구항 16

삭제

청구항 17

삭제

청구항 18

삭제

청구항 19

삭제

청구항 20

삭제

## 발명의 설명

### 기술 분야

[0001] 관련 출원들과의 상호 참조

[0002] 본 출원은 2019년 3월 1일에 출원된 국제 특허 출원 PCT/CN2019/076695의 우선권 및 혜택을 주장하는 2020년 3월 2일에 출원된 국제 특허 출원 PCT/CN2020/077415에 기초하고 있다. 상술한 모든 특허 출원은 그 전체가 참고로서 여기에 포함된다.

[0003] 본 출원은 비디오 코딩 및 디코딩 기법, 장치 및 시스템들에 관한 것이다.

### 배경 기술

[0004] 비디오 압축 기술의 발전에도 불구하고, 디지털 비디오는 여전히 인터넷과 다른 디지털 통신 네트워크 상의 최대 대역폭 사용(bandwidth use)을 차지한다. 비디오를 수신 및 재생(displaying)할 수 있는 연결된 사용 장치의 수가 증가함에 따라, 디지털 비디오 사용의 대역폭 요구는 계속 증가(grow)할 것이다.

### 발명의 내용

#### 해결하려는 과제

[0005] 이 명세서에서는 비디오 또는 화상을 디코딩 또는 인코딩하는 인트라 블록 복사 모드(intra block copy mode)를 위한 버퍼 관리(buffer management) 및 블록 벡터 코딩(block vector coding)의 다양한 실시예들과 기법들을 설명한다.

#### 과제의 해결 수단

[0006] 한 예시적 국면(aspect)에서, 비디오 혹은 이미지(시각 데이터) 처리 방법이 개시된다. 이 방법은 시각 매체 데이터의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드에서 예측을 위한 참조 샘플들을 저장하는 버퍼를 결정하고, 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생된 블록과 연관된 움직임 정보에 기초하는 인트라 블록 복사 모드에서 수행하는 단계, 현재 비디오 블록을 포함하고 블록 벡터 (BVx, Bvy)를 갖는 코딩 트리 유닛의 좌상단 위치에 대해 현재 비디오 블록의 위치 (x0, y0)에 공간적으로 위치한 샘플에 대해, 참조 위치 (P, Q)에서의 버퍼에서 대응되는 참조를 산출하고, 참조 위치 (P, Q)는 블록 벡터 (BVx, Bvy)를 이용하여 결정하는 단계, 참조 위치 (P, Q)가 버퍼 외부에 있다고 결정되면, 현재 비디오 블록을 포함하는 코딩 트리 유닛에 대한 현재 비디오 블록의 위치에 적어도 부분적으로 기초하여 참조 위치를 재산출하는 단계를 포함한다.

[0007] 다른 예시적 국면(aspect)에서, 다른 시각 데이터 처리 방법이 개시된다. 방법은 시각 매체 데이터의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드에서 예측을 위한 참조 샘플들을 저장하는 버퍼를 결정하는 단계를 포함하고, 여기서 변환은 현재 비디오 블록을 갖는 동일 비디오 영

역에 위치한 재생 블록과 연관된 움직임 정보에 연관된 움직임 정보를 기반으로 하는 인트라 블록 복사 모드에서 수행되고, 현재 비디오 블록을 포함하는 화면의 좌상 위치에 대한 현재 비디오 블록의 위치 (x, y)에 공간적으로 위치하고 블록 벡터(BVx, BVy)를 갖는 샘플에 대해, 현재 비디오 블록의 위치(x, y), 현재 비디오 블록의 차원, 화면의 차원, 현재 비디오 블록을 포함하는 코딩 트리 유닛의 차원, 혹은 버퍼의 차원 중 적어도 하나와 연관된 하나 이상의 조건을 만족하는 것에 적어도 부분적으로 기초하여 블록 벡터(Bvx, Bvy)를 할당하는 단계, 블록 벡터(BVx, Bvy)가 유효한지 확인하기 위한 검사 수행하는 단계, 블록 벡터 (BVx, Bvy)가 유효하다고 식별되면, 참조 위치 (P, Q)의 버퍼에서 대응되는 참조를 산출하는 단계를 포함하되, 참조 위치 (P, Q)는 블록 벡터 (BVx, Bvy), 위치 (x,y) 및 버퍼의 차원을 이용하여 결정된다.

[0008] 다른 예시적 국면(aspect)에서, 다른 시각 데이터 처리 방법이 개시된다. 방법은 시각 매체 데이터의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 현재 비디오 블록을 위한 블록 벡터 (BVx, BVy) 혹은 블록 벡터 차이(BVDx, BVDy)를 결정하는 단계를 포함하되, 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관된 움직임 정보에 기반하는 인트라 블록 복사 모드에서 수행되고, 블록 벡터(BVx, BVy)의 적어도 하나의 구성요소 혹은 블록 벡터 차이(BVDx, BVDy)의 적어도 하나의 구성요소가 범위 내에 놓이도록 정규화하는 단계를 포함한다.

[0009] 다른 예시적 국면(aspect)에서, 다른 시각 데이터 처리 방법이 개시된다. 방법은 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드에서 예측을 위한 재생 샘플들을 저장하는 데 사용되는 버퍼를 결정하는 단계를 포함하되, 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관된 움직임 정보에 기반하는 인트라 블록 복사 모드에서 수행되고, 순서에 따라 버퍼에 저장된 재생 샘플들을 업데이트 하는 단계를 포함한다.

[0010] 다른 예시적 국면(aspect)에서, 다른 시각 데이터 처리 방법이 개시된다. 방법은 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 수행하는 단계를 포함하되, 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관된 움직임 정보에 기반하는 인트라 블록 복사 모드에서 수행되고, 변환 동안, 예측 산출에 이용되는 제1 정밀도는 재생 산출에 이용되는 제2 정밀도보다 낮다.

[0011] 다른 예시적 국면(aspect)에서, 다른 시각 데이터 처리 방법이 개시된다. 방법은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관되는 움직임 정보에 기반하는 인트라 블록 복사 모드를 이용하여 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 수행하는 단계를 포함하되, 변환 동안  $nM \times nM$  크기의 참조 영역이 사용되고, n 및 M은 정수이고, 현재 비디오 블록은 코딩 트리 유닛 내에 위치하며, 참조 영역은 현재 비디오 블록에 대응되는 코딩 트리 유닛 행에서 가장 가까운 가용한  $n \times n$  코딩 트리 유닛들로부터의 샘플들을 포함한다.

[0012] 다른 예시적 국면(aspect)에서, 다른 시각 데이터 처리 방법이 개시된다. 방법은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생된 블록과 연관된 움직임 정보에 기초한 인트라 블록 복사 모드를 이용하여 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 수행하되, 변환 동안,  $nM \times pM$  크기인 참조 영역이 사용되되, n, p 및 M은 정수이고 현재 비디오 블록은 코딩 트리 유닛에 위치하고, 참조 영역은 현재 비디오 블록에 대응되는 코딩 트리 유닛 행에서 가장 가까운 가용한  $n \times p-1$  코딩 트리 유닛들로부터 샘플들을 포함한다.

[0013] 다른 예시적 국면(aspect)에서, 다른 시각 데이터 처리 방법이 개시된다. 방법은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생된 블록과 연관된 움직임 정보에 기초하는 인트라 블록 복사 모드를 이용하여 비디오 영역의 VPDU(virtual pipeline data unit)의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 수행하고, 변환 동안  $nM \times nM$  크기인 참조 영역이 사용되고, VPDU의 크기는  $kM \times kM$ 이고, 여기서 k, n 및 M은 정수이며 현재 비디오 블록은 코딩 트리 유닛에 위치하고, 참조 영역은 현재 비디오 블록에 대응되는 코딩 트리 유닛 행에서 가장 가까운 가용한  $n \times n-k$  코딩 트리 유닛들로부터 샘플들을 포함한다.

[0014] 다른 예시적 국면(aspect)에서, 다른 시각 데이터 처리 방법이 개시된다. 방법은 시각 매체 데이터의  $w \times h$  크기의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드에서 예측을 위한 참조 샘플들을 저장하는 버퍼를 결정하는 단계를 포함하고, 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생된 블록과 연관된 움직임 정보에 기반하는 인트라 블록 복사 모드에서 수행되고, 현재 비디오 블록을 포함하는  $M \times M$  크기인 코딩 트리 유닛(CTU)의 좌상단 위치에 관련된 현재 비디오 블록의 위치 (x0, y0)에 공간적으로 위치한 샘플에 대해, 버퍼에서 참조 위치(P, Q)에서 시작하여 대응되는 참조 영역을 산출하는 단계를 포함하되, 참조 위치(P, Q)는 블록 벡터 (BVx, BVy) 및/혹은 위치 (x0, y0)를 이용하여

결정되고, 비디오 영역을 갖는 참조 영역의 중복을 제한하기 위해 참조 영역 및/혹은 참조 위치(P, Q)에 하나 이상의 규칙 기반 제한 조건을 적용하는 단계를 포함한다.

[0015] 다른 예시적 국면(Aspect)에서, 다른 시각 데이터 처리 방법이 개시된다. 방법은 시각 매체 데이터의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드에서 예측을 위한 참조 샘플들을 저장하는 버퍼를 결정하는 단계를 포함하되, 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관된 움직임 정보에 기반하는 인트라 블록 복사 모드에서 수행되고, 현재 비디오 블록을 포함하는 코딩 유닛(CU)에 대한 현재 비디오 블록의 위치 (x0, y0)에 공간적으로 위치한 샘플에 대해 버퍼에서 참조 위치(P, Q)에서 시작하여 대응되는 참조 영역을 산출하는 단계 및 참조 영역 및 참조 위치를 조정하여 이전에 처리된 블록 중 예측에 사용되는 블록을 결정하는 단계를 포함한다.

[0016] 다른 예시적 국면(Aspect)에서, 다른 시각 데이터 처리 방법이 개시된다. 방법은 비디오의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 비디오의 구성요소 X를 이용하여 비디오의 구성요소 c의 현재 비디오 블록에 대응되는 블록 벡터의 유효성을 결정하는 단계를 포함하되, 구성요소 X는 비디오의 루마 구성요소와 상이하고, 현재 비디오 블록에 대해 블록 벡터가 유효하다는 결정 하에 블록 벡터를 이용하여 변환을 수행하는 단계를 포함하되, 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관된 움직임 정보를 기반으로 하는 인트라 블록 복사(IBC) 모드에서 변환이 수행된다.

[0017] 또 다른 예시적 국면에서, 전술한 하나의 방법을 구현하도록 구성된 프로세서를 포함하는 비디오 인코더 또는 디코더 장치가 개시된다.

[0018] 또 다른 예시적 국면에서, 컴퓨터 독출 가능한(computer readable) 프로그램 매체(program medium)가 개시된다. 이 매체는 개시된 방법들 중의 하나를 구현할 명령이 실행 가능한 프로세서로 유형화되는 코드를 저장한다.

[0019] 이들 및 다른 국면들이 이 명세서에 더 상세히 설명된다.

### 도면의 간단한 설명

[0020] 도 1은 현재 화면 참조 또는 인트라 블록 복사 비디오 또는 화상 코딩 기법의 예를 보인다.

도 2는 동적 기준 영역의 예를 보인다.

도 3은 (x,y)로부터 시작되는 블록의 코딩의 예를 보인다.

도 4는 이전에 코딩된 64x4 블록들을 선택하는 가능한 대체적 방법의 예들을 보인다.

도 5는 64x4 블록들의 코딩/디코딩 순서를 변경시키는 가능한 대체적 방법의 하나의 예를 보인다.

도 6은 비디오 또는 화상 처리의 예시적 방법의 흐름도.

도 7은 비디오 또는 화상 코딩 또는 디코딩을 위한 하드웨어 플랫폼의 블록도.

도 8은 64x64 블록들의 디코딩 순서가 상-하, 좌-우일 때 이전에 코딩된 64x4 블록들을 선택하는 다른 가능한 대체적 방법을 보인다.

도 9는 이전에 코딩된 64x4 블록들을 선택하는 다른 가능한 대체적 방법을 보인다.

도 10은 정형에 의한 디코딩 프로세스의 예시적 흐름도.

도 11은 64x64 블록들의 디코딩 순서가 좌-우, 상-하일 때 이전에 코딩된 64x4 블록들을 선택하는 다른 가능한 대체적 방법을 보인다.

도 12는 하나의 블록이 64x64 CTU를 나타낼 때, IBC 기준 버퍼 상태의 도면.

도 13은 IBC를 위한 기준 영역의 하나의 배열을 보인다.

도 14는 IBC를 위한 기준 영역의 다른 배열을 보인다.

도 15는 현재 가상 파이프라인 데이터 유닛(VPDU)이 화면 경계의 우측일 때 IBC를 위한 기준 영역의 다른 배열을 보인다.

도 16은 CTU 행 내의 VPDU들이 순차적으로 디코딩될 때 가상 버퍼의 상태의 예를 보인다.

도 17은 개시된 기법들이 구현될 수 있는 예시적 비디오 처리 시스템의 블록도.

도 18은 시각 데이터 처리의 예시적 방법을 보이는 흐름도.

도 19는 시각 데이터 처리의 예시적 방법을 보이는 흐름도.

도 20은 시각 데이터 처리의 예시적 방법을 보이는 흐름도.

도 21은 시각 데이터 처리의 예시적 방법을 보이는 흐름도.

도 22는 시각 데이터 처리의 예시적 방법을 보이는 흐름도.

도 23은 시각 데이터 처리의 예시적 방법을 보이는 흐름도.

도 24는 시각 데이터 처리의 예시적 방법을 보이는 흐름도.

도 25는 시각 데이터 처리의 예시적 방법을 보이는 흐름도.

도 26은 시각 데이터 처리의 예시적 방법을 보이는 흐름도.

도 27은 시각 데이터 처리의 예시적 방법을 보이는 흐름도.

도 28은 시각 데이터 처리의 예시적 방법을 보이는 흐름도.

### 발명을 실시하기 위한 구체적인 내용

[0021] 섹션의 표제들은 이 명세서에서 이해의 편의를 위한 것으로 각 섹션에 개시된 실시예들의 범위를 그 섹션에만 한정하는 것이 아니다. 이 명세서는 비디오 또는 화상(image)를 디코딩 또는 인코딩하는 인트라 블록 복사 모드(intra block copy mode)를 위한 버퍼 관리(buffer management) 및 블록 벡터 코딩(block vector coding)의 다양한 실시예들과 기법들을 설명한다.

#### [0022] 1. 요약(Summary)

[0023] 본 발명은 비디오 코딩 기술들에 관한 것이다. 더 구체적으로, 본 발명은 비디오 코딩에서 인트라 블록 복사에 관한 것이다. 본 발명은 예를 들어 다용도 비디오 코딩(Versatile Video Coding) 등의 개발 중인 표준에 적용될 수 있다. 본 발명은 미래 비디오 코딩 표준 또는 비디오 코덱에도 적용될 수 있다.

#### [0024] 2. 개요(Brief Discussion)

[0025] 비디오 코딩 표준들은 주로 잘 알려진 ITU-T 및 ISO/IEC의 개발을 통해 발전해 왔다. ITU-T는 H.261 및 H.263을 산출했고, ISO/IEC는 MPEG-1 및 MPEG-4 Visual을 산출했으며, 두 기관들은 공동으로 H.262/MPEG-2 Video와 H.264/MPEG-4 진보된 비디오 코딩(Advanced Video Coding; AVC) 및 H.265/HEVC 표준들을 산출했다. 비디오 코딩 표준인 H.262가 하이브리드 비디오 코딩 구조를 사용하므로 일시적 예측(temporal prediction) 더하기 변환 코딩(transform coding)이 사용된다. HEVC 너머의 미래 비디오 기술을 탐구하기 위해, 공동 비디오 탐구 팀(Joint Video Exploration Team; JVET)이 VCEG 및 MPEG에 의해 공동으로 2015년 설립되었다. 그 이후 많은 새로운 방법들이 JVET에 의해 채택되어 차세대 코딩 실험 모델(공동 탐구 모델; Joint Exploration Model; JEM)로 명명된 기준 소프트웨어에 투입되었다. 2018년 4월, VCEG (Q6/16)와 ISO/IEC JTC1 SC29/WG11(MPEG) 간의 공동 전문가 팀(Joint Video Expert Team; JVET)이 창설되어 HEVC에 비해 50% 비트전송률 저감을 목표로 하는 VVC 표준에 대한 연구를 하고 있다.

#### [0026] 2.1 HEVC/H.265에서의 인터 예측(Inter prediction in HEVC/H.265)

[0027] 각 인터 예측된(inter-predicted) PU는 하나 또는 2개의 참조 화면(reference picture) 목록들에 대한 모션 파라미터(motion parameter)들을 갖는다. 모션 파라미터들은 모션 벡터와 참조 화면 인덱스(index)를 포함한다. 2개의 참조 화면 목록들 중의 하나의 사용 역시 *inter\_pred\_idc*를 사용하여 신호될 수 있다. 모션 벡터들은 예측 변수(predictor)들에 대한 델타(delta)들로서 명시적으로(explicitly) 코딩될 수 있다.

[0028] CU가 스킵 모드(skip mode)로 코딩되면 하나의 PU가 이 CU에 연계되는데, 고, 중요 잔차계수(significant residual coefficient)가 없고, 코딩된 모션 벡터 델타 또는 참조 화면 인덱스도 없다. 병합 모드(merge mode)가 규정됨으로써 공간적(spatial) 및 시간적(temporal) 후보(candidate)들을 포함하여 현재 PU에 대한 모션 파라미터들이 이웃 PU들로부터 얻어진다. 병합 모드는 스킵 모드뿐 아니라 어떤 인터 예측된 PU에도 적용될 수 있다. 병합 모드에 대한 대안은 모션 파라미터들의 명시적 전송(explicit transmission)인데, 여기서 각 참조

화면 목록에 대한 참조 화면의 참조 화면 인덱스에 대응하는 모션 벡터(더 정확히는 모션 벡터 예측 변수와 비교한 모션 벡터 차이(motion vector differences; MVD)와 참조 화면 목록 사용(usage)이 각 PU 당 명시적으로 신호된다. 이러한 모드는 이 명세서에서 진보된 모션 벡터 예측(Advanced motion vector prediction; AMVP)로 명명된다.

[0029] 시그널링(signaling)이 2개의 참조 화면 목록들 중의 하나가 사용될 것으로 표시(indicate)하면, 샘플들의 한 블록으로부터 PU가 산출된다. 이는 "단 예측(uni-prediction)"으로 지칭된다. 단 예측은 P 슬라이스(P-slice) 및 B 슬라이스들 모두에 적용 가능하다.

[0030] 시그널링이 참조 화면 목록들 양자를 사용할 것을 표시하면, 샘플들의 2개의 블록들로부터 PU가 산출된다. 이는 "양 예측(bi-prediction)"으로 지칭된다. 양 예측은 B 슬라이스들에만 적용 가능하다.

[0031] 이하의 텍스트는 HEVC에 규정된 인터 예측 모드(inter prediction mode)에 대한 상세를 제공한다. 설명은 병합 모드와 함께 시작된다.

## [0032] 2.2 현재 화면 참조(Current Picture Referencing)

[0033] 한때 인트라 블록 복사(Intra Block Copy; IBC)로 명명되었던 현재 화면 참조(Current Picture Referencing; CPR)가 HEVC 스크린 콘텐츠 코딩 확장(Screen Content Coding extensions)(HEVC-SCC)과 VVC 시험 모델에 채택되었다. IBC는 인터 프레임 코딩(inter-frame coding)으로부터 인트라 프레임 코딩(intra-frame coding)으로 모션 보상(motion compensation)의 개념을 확장하였다. 도 1에 도시된 바와 같이, CPR이 적용될 때 현재 블록은 동일 화면 내의 기준 블록에 의해 예측된다. 기준 블록 내의 샘플들은 현재 블록이 코딩 또는 디코딩되기 전에 이미 재생(reconstructed)되어 있어야 한다. CPR이 카메라 캡처된(camera-captured) 시퀀스들에 그렇게 효율적은 아니지만, 이는 스크린 콘텐츠(screen content)에 대한 상당한 코딩 게인(coding gain)을 보인다. 그 이유는 스크린 콘텐츠 화면 내에 아이콘 과 문자(text character) 등 수많은 반복 패턴들이 존재하기 때문이다. CPR은 이러한 반복 패턴들 간의 리던던시(redundancy)를 효과적으로 제거할 수 있다. HEVC-SCC에서, 현재 화면을 그 참조 화면들로 선택한다면 인터 코딩된(inter-coded) 코딩 유닛(coding unit; CU)에 CPR을 적용할 수 있다. 이 경우 MV는 블록 벡터(BV)로 개명되고, BV는 항상 정수 화소(integer-pixel) 정밀도(precision)를 갖는다. 메인 프로파일(main profile)인 HEVC와 호환(compatible with)을 위해, 현재 화면은 디코딩된 화면 버퍼(Decoded Picture Buffer; DPB) 내에 "장기(long-term)" 참조 화면으로 마킹된다. 마찬가지로 다시점/3D 비디오 코딩 표준(multiple view/3D video coding standard)들에서 인터 시점(inter-view) 참조 화면 역시 "장기" 참조 화면으로 마킹됨에 유의해야 한다.

[0034] BV가 그 기준 블록을 찾은 다음, 기준 블록을 복사함으로써 예측이 생성될 수 있다. 기준 화소들을 원래 신호들로부터 차감함으로써 잔차(residual)가 얻어질 수 있다. 이어서 다른 코딩 모드들에서와 같이 변환과 양자화(quantization)가 적용될 수 있다.

[0035] 도 1은 현재 화면 참조(Current Picture Referencing)의 예시적 도면이다.

[0036] 그러나 기준 블록이 화면 외부에 있거나, 현재 블록과 중첩(overlap)되거나, 재생 영역(reconstructed area) 외부이거나, 어떤 제약들에 의해 제한된 유효 영역(valid area) 외부라면, 일부 또는 전부의 화소값들이 정의되지 않는다. 기본적으로, 이러한 문제를 해결하는 2가지 해법들이 있다. 하나는 예를 들어 비트스트림 적합성(bitstream conformance)에서 이러한 상황을 허용하지 않는 것이다. 다른 하나는 이 정의되지 않은 화소값들에 패딩(padding)을 적용하는 것이다. 다음 세부 섹션들에서 이 해법들을 상세히 설명한다.

## [0037] 2.3 HEVC 스크린 콘텐츠 코딩 확장에서의 CPR(CPR in HEVC Screen Content Coding extensions)

[0038] HEVC의 스크린 콘텐츠 코딩 확장에서, 블록이 현재 화면을 기준으로 사용한다면, 다음 스펙 텍스트(spec text)에서 지시된 바와 같이 전체 기준 블록이 사용 가능한 재생 영역 이내일 것이 보장되어야 하는데:

The variables offsetX and offsetY are derived as follows:

$$\text{offsetX} = (\text{ChromaArrayType} == 0) ? 0 : (\text{mvCLX}[0] \& 0x7 ? 2 : 0) \quad (8-104)$$

$$\text{offsetY} = (\text{ChromaArrayType} == 0) ? 0 : (\text{mvCLX}[1] \& 0x7 ? 2 : 0) \quad (8-105)$$

It is a requirement of bitstream conformance that when the reference picture is the current picture, the luma motion vector mvLX shall obey the following constraints:

- When the derivation process for z-scan order block availability as specified in clause 6.4.1 is invoked with (xCurr, yCurr) set equal to (xCb, yCb) and the neighbouring luma location (xNbY, yNbY) set equal to (xPb + (mvLX[0] >> 2) - offsetX, yPb + (mvLX[1] >> 2) - offsetY) as inputs, the output shall be equal to TRUE.
- When the derivation process for z-scan order block availability as specified in clause 6.4.1 is invoked with (xCurr, yCurr) set equal to (xCb, yCb) and the neighbouring luma location (xNbY, yNbY) set equal to (xPb + (mvLX[0] >> 2) + nPbW - 1 + offsetX, yPb + (mvLX[1] >> 2) + nPbH - 1 + offsetY) as inputs, the output shall be equal to TRUE.
- One or both of the following conditions shall be true:
  - The value of (mvLX[0] >> 2) + nPbW + xB1 + offsetX is less than or equal to 0.
  - The value of (mvLX[1] >> 2) + nPbH + yB1 + offsetY is less than or equal to 0.
- The following condition shall be true:

$$(\text{xPb} + (\text{mvLX}[0] \gg 2) + \text{nPbSw} - 1 + \text{offsetX}) / \text{CtbSizeY} - \text{xCb} / \text{CtbSizeY} \leq \text{yCb} / \text{CtbSizeY} - (\text{yPb} + (\text{mvLX}[1] \gg 2) + \text{nPbSh} - 1 + \text{offsetY}) / \text{CtbSizeY} \quad (8-106)$$

[0039]

[0040]

이에 따라, 기준 블록이 현재 블록과 중첩되거나 기준 블록이 화면 외부인 상황은 발생되지 않을 것이다. (그러므로) 기준 또는 예측 블록을 패딩할(pad)할 필요가 없다.

[0041]

## 2.4 CPR/IBC의 예들(EXAMPLES of CPR/IBC)

[0042]

VVC 시험 모델에서, 전체 기준 블록은 현재 코딩 트리 유닛(current coding tree unit; CTU)을 가져야 하고 현재 블록과 중첩되지 않아야 한다. 이에 따라 기준 또는 예측 블록에 패딩할 필요가 없다.

[0043]

이중 트리(dual tree)가 이네이블되면(enabled) 분할 구조(partition structure)는 루마(luma)로부터 크로마(chroma) CTU들까지 다를 수 있다. 그러므로 4:2:0 칼라 포맷에 대해 (예를 들어 CU 등의) 하나의 크로마 블록은 복수의 루마 CTU들로 분할된 하나의 병치된(collocated) 루마 영역에 대응할 수 있다.

[0044]

크로마 블록은 다음 조건들이 참(true)일 때만 CPR 모드로 코딩될 수 있다:

[0045]

1) 병치된 루마 블록 내의 각 루마 CU가 CPR 모드로 코딩되어야 하고

[0046]

2) 각 루마 4x4 블록의 BV가 먼저 크로마 블록의 BV로 변환되고 크로마 블록의 BV가 유효한(valid) BV이다.

[0047]

2개의 조건들 중의 어느 것이 거짓(false)이면, 크로마 블록은 CPR 모드로 코딩될 수 없다.

[0048]

'유효한 BV(valid BV)'의 정의가 다음 제약들을 가짐에 유의해야 하는데:

[0049]

1) BV에 의해 식별된 기준 블록 내의 모든 샘플들은 제한된 탐색 범위(search range)를 가져야 한다(예를 들어 현재 VVC 설계에서 동일한 CTU 내에 있어야 한다).

[0050]

2) BV에 의해 식별된 기준 블록 내의 모든 샘플들은 재생(reconstructed)되었어야 한다.

[0051]

## 2.5 CPR/IBC의 예들(EXAMPLES of CPR/IBC)

[0052]

일부 예들에서, CPR/IBC를 위한 기준 영역은 현재 CTU로 제한되는데, 이는 128x128까지이다. 기준 영역은 CPR/IBC를 위한 참조 샘플들을 저장하는 메모리를 재사용하기 위해 동적으로 변경되어, CPR/IBC를 위한 기준 버퍼(reference buffer)가 하나의 CTU를 유지하거나 이로부터 축소되면서 CPR/IBC 블록이 더 많은 기준 후보(reference candidate)를 가질 수 있다.

[0053]

도 2는 하나의 방법을 보이는데, 여기서 블록은 64x64이고 CTU는 64x64 블록들을 포함한다. 64x64 블록을 코딩할 때 이전 3개의 64x64 블록들이 기준으로 사용될 수 있다. 그러므로써 디코더는 CPR/IBC를 지원하는 데 정확히 4개의 64x64 블록들만을 저장할 필요가 있다.

[0054]

화면의 상-좌(upper-left) 모서리에 대한 현재 루마 CU의 위치가 (x, y)이고 블록 벡터가 (BVx, BVy)라고 가정하자. 현재의 설계에서, 루마 위치 ((x+BVx)>>6<<6+(1<<7))라는 것으로 BV가 유효하다고 할 수 있다면,

$(y+Bv_y) \gg 6 \ll 6$ 는 재생되지 않았고  $((x+Bv_x) \gg 6 \ll 6 + (1 \ll 7))$ ,  $(y+Bv_y) \gg 6 \ll 6$ 은  $(x \gg 6 \ll 6, y \gg 6 \ll 6)$ 과 같지 않다.

[0055] **2.6 루프-내 정형(In-loop reshaping; ILR)**

[0056] 루프-내 정형(in-loop reshaping; ILR)의 기본적 아이디어는 (제1 도메인의) 원래의 신호(예측/재생 신호)를 제2 도메인(정형 도메인)으로 변환하는 것이다.

[0057] 루프-내 루마 정형기(in-loop luma reshapers)는 한 쌍의 룩업 테이블(look-up tables; LUTs)들로 구현되지만, 다른 것은 신호된 LUT로부터 연산될 수 있으므로 2개의 LUT들 중 하나만이 신호될 필요가 있다. 각 LUT는 1차원, 10비트, 1024-입력(entry) 매핑 테이블(mapping table)(1D-LUT)이다. 하나의 LUT는 입력 루마 코드 값( $Y_i$ )을 변경된 값  $Y_r$ :  $Y_r = \text{InvLUT}[Y_i]$ 로 매핑하는(map) 정방향(forward) LUT( $\text{FwdLUT}$ )이다. 다른 LUT는 변경된 코드 값( $Y_i$ )을  $\hat{Y}_i$ :  $\hat{Y}_i = \text{InvLUT}[Y_r]$ 로 매핑하는 역방향(inverse) LUT( $\text{InvLUT}$ )이다( $\hat{Y}_i$ 는  $Y_i$ 의 재생 값을 나타낸다).

[0058] **2.6.1 PWL 모델(PWL model)**

[0059] 개념적으로 조각별 선형(piece-wise linear; PWL)은 다음 방법으로 구현되는데:

[0060]  $x_1, x_2$ 가 2개의 입력 회동점(input pivot point)들이고,  $y_1, y_2$ 가 한 조각(piece)을 위한 그 대응 출력 회동점들이고 가정한다.  $x_1$ 과  $x_2$  사이의 어떤 입력값( $x$ )에 대한 출력값  $y$ 는 다음 식으로 보간될(interpolated) 수 있는데:

[0061] 
$$y = ((y_2 - y_1) / (x_2 - x_1)) * (x - x_1) + y_1$$

[0062] 고정 소수점(fixed point) 구현에서는 이 식이 다음으로 수정될 수 있는데:

[0063] 
$$y = ((m * x + 2^{\text{FP\_PREC}-1}) \gg \text{FP\_PREC}) + c$$

[0064] 여기서  $m$ 은 스칼라(scalar),  $c$ 는 오프셋(offset), 그리고  $\text{FP\_PREC}$ 는 정밀도를 규정하는 상수 값이다.

[0065] 일부 예들에서, PWL 모델은 1024 입력  $\text{FwdLUT}$  및  $\text{InvLUT}$  매핑 테이블을 사전 연산(precompute)하는 데 사용되지만; PWL 모델은 또한 LUT들을 사전 연산하지 않고 동일한 현장(on-the-fly) 매핑 값의 연산의 구현 역시 가능하게 한다.

[0066] **2.6.2.1 루마 정형(Luma reshaping)**

[0067] 루프-내 루마 정형 방법은 인터 슬라이스 재생에서의 블록별 인트라 예측에 대한 디코딩 지연(latency) 역시 제거하는 더 낮은 복잡성(complexity)의 파이프라인을 제공한다. 인트라 예측은 인터 및 인트라 슬라이스들 양자에 대한 정형 도메인(reshaped domain)에서 수행된다.

[0068] 인트라 예측은 슬라이스 종류(type)에 무관하게 정형 도메인에서 수행된다. 이러한 배열에서, 인트라 예측은 이전 TU 재생이 수행된 바로 다음에 시작될 수 있다. 이러한 배열은 또한 슬라이스 종속(dependent) 대신에 인트라 모드에 대한 통합 프로세스(unified process)를 제공한다. 도 10은 모드에 기반한 CE12-2 디코딩 프로세스의 블록도를 보인다.

[0069] 16 조각의 조각별 선형(PWL) 모델들은 32 조각 PWL 모델 대신 루마 및 크로마 잔차 스케일링(residue scaling)에 대해 시험된다.

[0070] 루프-내 루마 정형기를 사용한 인터 슬라이스 재생(연속색 음영의 블록들은 정형 도메인(reshaped domain): 루마 잔차(luma residue); 예측된 인트라 루마; 및 재생된 인트라 루마 내의 신호를 표시한다).

[0071] **2.6.2.2 루마 종속 크로마 잔차 스케일링(Luma-dependent chroma residue scaling)**

[0072] 루마 종속 크로마 잔차 스케일링(luma-dependent chroma residue scaling)은 고정 소수점 정수 연산(operation)으로 구현되는 곱셈 프로세스(multiplicative process)이다. 크로마 잔차 스케일링은 크로마 신호와 루마 신호의 상호작용을 보상한다. 크로마 잔차 스케일링은 TU 레벨(level)에 적용된다. 더 구체적으로 다음이 적용된다:

[0073] - 인트라에 대해, 재생 루마가 평균화된다.

- [0074] - 인터에 대해, 예측 루마가 평균화된다.
- [0075] 이 평균은 PWL 모델 내의 인덱스(index)를 식별하는 데 사용된다. 이 인덱스는 척도 인자(scaling factor; cScaleInv)를 식별한다. 크로마 잔차가 이 수로 곱해진다.
- [0076] 크로마 척도 인자는 재생 루마 값이 아니라 정방향 매핑된(forward-mapped) 예측 루마 값으로부터 연산되는 것에 유의해야 한다.
- [0077] **2.6.2.3 ILR 측 정보의 시그널링(Signalling of ILR side information)**
- [0078] 파라미터들은 (ALF와 유사한) 타일 그룹 헤더(tile group header) 내에 (지금) 전송된다. 이는 40-100 비트가 소용된다고 알려졌다.
- [0079] 일부 예들에서, 추가된 신택스(syntax)는 이탤릭체로 강조된다.
- [0080] <7.3.2.1에서 시퀀스 파라미터 세트 RBSP 신택스>

표 1

| seq_parameter_set_rbsp() {                               | Descriptor |
|----------------------------------------------------------|------------|
| <b>sps_seq_parameter_set_id</b>                          | · ue(v)    |
| · · ·                                                    | ·          |
| <b>sps_triangle_enabled_flag</b>                         | · u(1)     |
| <b>sps_ladf_enabled_flag</b>                             | · u(1)     |
| if (sps_ladf_enabled_flag) {                             | ·          |
| <b>sps_num_ladf_intervals_minus2</b>                     | · u(2)     |
| <b>sps_ladf_lowest_interval_qp_offset</b>                | · se(v)    |
| for(i = 0; i < sps_num_ladf_intervals_minus2 + 1; i++) { | ·          |
| <b>sps_ladf_qp_offset[i]</b>                             | · se(v)    |
| <b>sps_ladf_delta_threshold_minus1[i]</b>                | · ue(v)    |
| }                                                        | ·          |
| }                                                        | ·          |
| <b>sps_reshaper_enabled_flag</b>                         | · u(1)     |
| rbbsp_trailing_bits()                                    | ·          |
| }                                                        | ·          |

- [0082] <7.3.3.1에서 일반적 타일 그룹 헤더 신택스(General tile group header syntax)>

표 2

| tile_group_header() {                                                                           | Descriptor |
|-------------------------------------------------------------------------------------------------|------------|
| · · ·                                                                                           |            |
| if(num_tiles_in_tile_group_minus1 > 0) {                                                        |            |
| <b>offset_len_minus1</b>                                                                        | ue(v)      |
| for(i = 0; i < num_tiles_in_tile_group_minus1; i++)                                             |            |
| <b>entry_point_offset_minus1[i]</b>                                                             | u(v)       |
| }                                                                                               |            |
| if (sps_reshaper_enabled_flag) {                                                                | ·          |
| <b>tile_group_reshaper_model_present_flag</b>                                                   | · u(1)     |
| if (tile_group_reshaper_model_present_flag)                                                     | ·          |
| tile_group_reshaper_model ()                                                                    | ·          |
| <b>tile_group_reshaper_enable_flag</b>                                                          | · u(1)     |
| if (tile_group_reshaper_enable_flag && (!(qtbtt_dual_tree_intra_flag && tile_group_type == I))) | ·          |
| <b>tile_group_reshaper_chroma_residual_scale_flag</b>                                           | · u(1)     |
| }                                                                                               | ·          |
| byte_alignment()                                                                                | ·          |
| }                                                                                               | ·          |

[0084] <새로운 선택스 테이블 타일 그룹 정형기 모델 추가:>

표 3

| <i>tile_group_reshaper_model () {</i>                                                | <i>Descriptor</i> |
|--------------------------------------------------------------------------------------|-------------------|
| <i>reshaper_model_min_bin_idx</i>                                                    | <i>ue(v)</i>      |
| <i>reshaper_model_delta_max_bin_idx</i>                                              | <i>ue(v)</i>      |
| <i>reshaper_model_bin_delta_abs_cw_prec_minus1</i>                                   | <i>ue(v)</i>      |
| <i>for (i = reshaper_model_min_bin_idx; i &lt;= reshaper_model_max_bin_idx; i++)</i> |                   |
| <i>{</i>                                                                             |                   |
| <i>    reshape_model_bin_delta_abs_CW [i]</i>                                        | <i>u(v)</i>       |
| <i>    if (reshaper_model_bin_delta_abs_CW[i]) &gt; 0)</i>                           |                   |
| <i>        reshaper_model_bin_delta_sign_CW_flag[i]</i>                              | <i>u(1)</i>       |
| <i>    }</i>                                                                         |                   |
| <i>}</i>                                                                             |                   |

[0086] 일반적 시퀀스 파라미터 세트 RBSP 시맨틱(semantic)들에 다음 시맨틱들을 추가하는데:

[0087] 1인 **sps\_reshaper\_enabled\_flag** 은 정형기가 코딩된 비디오 시퀀스(coded video sequence; CVS)에 사용된다고 규정한다. 0인 **sps\_reshaper\_enabled\_flag**은 정형기가 CVS에 사용되지 않는다고 규정된다. **tile\_group\_reshaper\_model\_present\_flag**이 존재하지 않으면 이는 0이라는 것을 의미한다.

[0088] 1인 **tile\_group\_reshaper\_enabled\_flag**은 정형기가 현재 타일 그룹에 대해 이네이블되었다고 규정한다. 0인 **tile\_group\_reshaper\_enabled\_flag**은 정형기가 현재 타일 그룹에 대해 이네이블되지 않았다고 규정한다. **tile\_group\_reshaper\_enable\_flag**이 존재하지 않으면 이는 0이라는 것을 의미한다.

[0089] 1인 **tile\_group\_reshaper\_chroma\_residual\_scale\_flag**은 크로마 잔차 스케일링(chroma residual scaling)이 현재 타일 그룹에 대해 이네이블되었다고 규정한다. 0인 **tile\_group\_reshaper\_chroma\_residual\_scale\_flag**은 크로마 잔차 스케일링이 현재 타일 그룹에 대해 이네이블되지 않았다고 규정한다. **tile\_group\_reshaper\_chroma\_residual\_scale\_flag**이 존재하지 않으면 이는 0이라는 것을 의미한다.

[0090] **tile\_group\_reshaper\_model()** 선택스를 추가한다

[0091] **reshape\_model\_min\_bin\_idx**는 정형기 구축(construction) 프로세스에 사용될 최소 빈(bin)(또는 조각) 인덱스를 규정한다. **reshape\_model\_min\_bin\_idx**의 값은 양단을 포함하여(inclusive) 0 내지 **MaxBinIdx**의 범위에 있어야 한다. **MaxBinIdx**의 값은 15여야 한다.

[0092] **reshape\_model\_delta\_max\_bin\_idx**는 정형기 구축 프로세스에 사용될 최대 허용 빈(또는 조각) 인덱스(**MaxBinIdx**) 빼기 최대 빈 인덱스를 규정한다. **reshape\_model\_max\_bin\_idx**의 값은 **MaxBinIdx** - **reshape\_model\_delta\_max\_bin\_idx**으로 설정된다.

[0093] **reshaper\_model\_bin\_delta\_abs\_cw\_prec\_minus1** 더하기 1은 선택스 **reshape\_model\_bin\_delta\_abs\_CW[i]**의 표현에 사용되는 비트 수를 규정한다.

[0094] **reshape\_model\_bin\_delta\_abs\_CW[i]**는 i번째 빈에 대한 절대 델타 코드워드(codeword) 값을 규정한다.

[0095] **reshaper\_model\_bin\_delta\_sign\_CW\_flag[i]**는 **reshape\_model\_bin\_delta\_abs\_CW[i]**의 부호를 다음과 같이 규정한다:

[0096] - **reshape\_model\_bin\_delta\_sign\_CW\_flag[i]**가 0이면 대응 변수 **RspDeltaCW[i]**는 양의 값이다.

[0097] - 그렇지 않으면(**reshape\_model\_bin\_delta\_sign\_CW\_flag[i]**가 0이 아니면) 대응 변수 **RspDeltaCW[i]**는 음의 값이다.

[0098] **reshape\_model\_bin\_delta\_sign\_CW\_flag[i]**이 존재하지 않으면 0임을 의미한다.

[0099] 변수 **RspDeltaCW[i] = (1 - 2\*reshape\_model\_bin\_delta\_sign\_CW[i]) \* reshape\_model\_bin\_delta\_abs\_CW [i];**

[0100] 변수 RspCW[i]는 다음과 같이 도출되는데:

[0101] 변수 OrgCW 가  $(1 \ll \text{BitDepth}_Y) / (\text{MaxBinIdx} + 1)$ 으로 설정된다.

[0102] -  $\text{reshaper\_model\_min\_bin\_idx} \leq i \leq \text{reshaper\_model\_max\_bin\_idx}$ 이면  $\text{RspCW}[i] = \text{OrgCW} + \text{RspDeltaCW}[i]$ .

[0103] - 그렇지 않으면,  $\text{RspCW}[i] = 0$ .

[0104]  $\text{BitDepth}_Y$ 의 값이 10이면  $\text{RspCW}[i]$ 의 값은  $32$  내지  $2 * \text{OrgCW} - 1$ 의 범위에 있어야 한다.

[0105]  $i$ 가 양단을 포함하여  $0$  내지  $\text{MaxBinIdx} + 1$ 의 범위에 있는 변수  $\text{InputPivot}[i]$ 는 다음과 같이 도출되는데:

[0106]  $\text{InputPivot}[i] = i * \text{OrgCW}$

[0107]  $i$ 가 양단을 포함하여  $0$  내지  $\text{MaxBinIdx} + 1$ 의 범위에 있는 변수  $\text{ReshapePivot}[i]$ 와,  $i$ 가 양단을 포함하여  $0$  내지  $\text{MaxBinIdx} + 1$ 의 범위에 있는 변수  $\text{ScaleCoef}[i]$  및  $\text{InvScaleCoeff}[i]$ 는 다음과 같이 도출되는데:

[0108]  $\text{shiftY} = 14$

[0109]  $\text{ReshapePivot}[0] = 0;$

[0110]  $\text{for}(i = 0; i \leq \text{MaxBinIdx}; i++) \{$

[0111]          $\text{ReshapePivot}[i+1] = \text{ReshapePivot}[i] + \text{RspCW}[i]$

[0112]          $\text{ScaleCoef}[i] = (\text{RspCW}[i] * (1 \ll \text{shiftY}) + (1 \ll (\text{Log2}(\text{OrgCW}) - 1))) \gg (\text{Log2}(\text{OrgCW}))$

[0113]          $\text{if} (\text{RspCW}[i] == 0)$

[0114]                  $\text{InvScaleCoeff}[i] = 0$

[0115]          $\text{else}$

[0116]                  $\text{InvScaleCoeff}[i] = \text{OrgCW} * (1 \ll \text{shiftY}) / \text{RspCW}[i]$

[0117]          $\}$

[0118]  $i$ 가 양단을 포함하여  $0$  내지  $\text{MaxBinIdx} + 1$ 의 범위에 있는 변수  $\text{ChromaScaleCoef}[i]$ 는 다음과 같이 도출되는데:

[0119]  $\text{ChromaResidualScaleLut}[64] = \{16384, 16384, 16384, 16384, 16384, 16384, 16384, 8192, 8192, 8192, 8192, 5461, 5461, 5461, 5461, 4096, 4096, 4096, 4096, 3277, 3277, 3277, 3277, 2731, 2731, 2731, 2731, 2341, 2341, 2341, 2048, 2048, 2048, 1820, 1820, 1820, 1638, 1638, 1638, 1638, 1489, 1489, 1489, 1489, 1365, 1365, 1365, 1260, 1260, 1260, 1260, 1170, 1170, 1170, 1170, 1092, 1092, 1092, 1092, 1024, 1024, 1024, 1024\};$

[0120]  $\text{shiftC} = 11$

[0121] -  $(\text{RspCW}[i] == 0)$ 이면

[0122]  $\text{ChromaScaleCoef}[i] = (1 \ll \text{shiftC})$

[0123] - 그렇지 않으면  $(\text{RspCW}[i] \neq 0)$ ,  $\text{ChromaScaleCoef}[i] = \text{ChromaResidualScaleLut}[\text{RspCW}[i] \gg 1]$

[0124] **2.6.2.4 ILR의 사용(Usage of ILR)**

[0125] 인코더 측에서, 각 화면(또는 타일 그룹)은 먼저 정형 도메인으로 변환된다. 그리고 모든 코딩 프로세스가 정형 도메인에서 수행된다. 인트라 예측에 대해서는 이웃 블록들이 정형 도메인 내에 있고; 인터 예측에 대해서는 (원래 도메인의 디코딩된 화면 버퍼(decoded picture buffer)에서 생성된) 기준 블록들이 먼저 정형 도메인으로 변환된다. 이어서 잔차가 생성되어 비트스트림에 코딩된다.

[0126] 전체 화면(또는 타일 그룹)이 인코딩/디코딩을 완료하면, 정형 도메인 내의 샘플들이 원래 도메인으로 변환되고, 이어서 블록해제 필터(deblocking filter)와 다른 필터들이 적용된다.

[0127] 예측 신호에 대한 정방향(forward) 정형은 다음 경우에 디스에이블되는데:

- [0128] 현재 블록이 인트라 코딩되었을(intra-coded) 경우
- [0129] 현재 블록이 CPR(현재 화면 참조, 일명 인트라 블록 복사(IBC))로 코딩된 경우.
- [0130] 현재 블록은 인터-인트라 조합 모드(combined inter-intra mode; CIIP)로 코딩되고 정방향 정형은 인트라 예측 블록에 대해 디스에이블된다.
- [0131] **3. 여러 실시예들에 의해 해결되는 문제들의 예들(Examples of problems solved by various embodiments)**
- [0132] CPR/IBC의 현재 설계에는 몇 가지 문제들이 존재한다.
- [0133] 1) 기준 영역들이 동적으로 변화하는데, 이것이 인코더/디코더 처리를 복잡하게 한다.
- [0134] 2) 유효하지 않은(invalid) 블록 벡터들이 쉽게 생성되고 확인이 어려운데, 이것이 인코더와 디코더 양자를 복잡하게 한다.
- [0135] 3) 불규칙한(irregular) 기준 영역이 블록 벡터의 비효율적인 코딩을 야기한다.
- [0136] 4) 128x128보다 작은 CTU 크기를 어떻게 취급할지가 명확하지 않다.
- [0137] 5) 크로마 블록들에 대해 BV가 유효한지 유효하지 않은지 여부의 판단 프로세스에서, 이 판단이 루마 샘플들의 사용 가능성에 기반하는데, 이중 트리 구조 구조에 의해 잘못된 판단으로 결과될 수 있다.
- [0138] **4. 예시적 실시예들(Example embodiments)**
- [0139] 일부 실시예들에서, 규칙적 버퍼(regular buffer)가 CPR/IBC 블록에 사용되어 기준을 얻을 수 있다.
- [0140] 함수  $isRec(x,y)$ 는 화소(x,y)가 재생되어 IBC 모드에 참조되었는지를 표시하도록 정의된다. (x, y)가 화면 밖의 다른 슬라이스/타일/브릭(brick)에 있으면  $isRec(x,y)$ 가 거짓을 회신(return)하고; (x, y)가 재생되지 않았으면  $isRec(x,y)$ 가 거짓을 회신한다. 다른 예에서, 샘플(x, y)이 재생되었지만 기준 영역 밖이거나/다른 VPDU에 존재하는 등 어떤 다른 조건들이 충족되면, 역시 사용 불가능한 것으로 마킹하고,  $isRec(x,y)$ 가 거짓을 회신한다.
- [0141] 함수  $isRec(c, x,y)$ 는 컴포넌트(component) c에 대한 샘플(x, y)이 사용가능한지를 표시하도록 정의된다. 예를 들어, 샘플(x, y)이 아직 재생되지 않았다면 이는 사용 불가능하다고 마킹된다. 다른 예에서, 샘플(x, y)이 재생되었지만 기준 영역 밖이거나/다른 슬라이스/브릭에 있거나/허용된 기준 영역 밖의 다른 VPDU에 존재하는 등 어떤 다른 조건들이 충족되면, 역시 사용 불가능한 것으로 마킹된다. 샘플(x, y)이 사용 불가능하면  $isRec(c, x,y)$ 는 거짓을 회신하고, 그렇지 않으면 참을 회신한다.
- [0142] 이하의 논의에서는 참조 샘플이 재생 샘플이 될 수 있다. '화소 버퍼(pixel buffer)'가 '단일 칼라 컴포넌트의 버퍼(buffer of one color component)' 또는 '다중 칼라 컴포넌트들의 버퍼(buffer of multiple color components)'에 응답할 수 있음에 유의해야 한다.
- [0143] **CPR/IBC에 대한 기준 버퍼(Reference buffer for CPR/IBC)**
- [0144] 1. CPR/IBC에 대한 루마 참조 샘플들을 저장하기 위해 MxN 화소 버퍼를 사용할 것이 제안된다.
- [0145] a. 하나의 예에서, 버퍼 크기는 64x64이다.
- [0146] b. 하나의 예에서, 버퍼 크기는 128x128이다.
- [0147] c. 하나의 예에서, 버퍼 크기는 64x128이다.
- [0148] d. 하나의 예에서, 버퍼 크기는 128x64이다.
- [0149] e. 하나의 예에서 M은 CTU의 높이와 같다.
- [0150] f. 하나의 예에서  $N=nH$ 인데, 여기서 H는 CTU의 높이이고, n은 양의 정수이다.
- [0151] g. 하나의 예에서, M은 CTU의 폭과 같다.
- [0152] h. 하나의 예에서,  $M=mWd1sep$ , 여기서 W는 CTU의 폭이고, m은 양의 정수이다.
- [0153] i. 하나의 예에서, 버퍼 크기는 96x128 또는 128x96 등 CTU 크기와 같지 않다.
- [0154] j. 하나의 예에서, 버퍼 크기는 CTU 크기와 같다.

- [0155] k. 하나의 예에서,  $M=mW$  및  $N=H$ 인데, 여기서  $W$  및  $H$ 는 CTU의 폭과 높이이고,  $m$ 은 양의 정수이다.
- [0156] l. 하나의 예에서  $M=W$  및  $N=nH$ 인데, 여기서  $W$  및  $H$ 는 CTU의 폭과 높이이고,  $n$ 은 양의 정수이다.
- [0157] m. 하나의 예에서,  $M=mW$  및  $N=nH$ 인데, 여기서  $W$  및  $H$ 는 CTU의 폭과 높이이고,  $m$  및  $n$ 은 양의 정수이다.
- [0158] n. 이상의 예에서,  $m$  및  $n$ 은 CTU 크기에 좌우될 수 있다.
- [0159] I. 하나의 예에서 CTU 크기가 128x128일 때  $m=1$  및  $n=1$ 이다.
- [0160] ii. 하나의 예에서 CTU 크기가 64x64일 때  $m=4$  및  $n=1$ 이다.
- [0161] iii. 하나의 예에서 CTU 크기가 32x32일 때  $m=16$  및  $n=1$ 이다.
- [0162] iv. 하나의 예에서 CTU 크기가 16x16일 때  $m=64$  및  $n=1$ 이다.
- [0163] o. 이와는 달리, 버퍼 크기가 CTU 크기에 대응한다.
- [0164] p. 이와는 달리, 버퍼 크기가 가상 파이프라인 데이터 유닛(Virtual Pipeline Data Unit; VPDU) 크기에 대응한다.
- [0165] q.  $M$  및/또는  $N$ 은 VPS/SPS/PPS/화면 헤더/슬라이스 헤더/타일 그룹 헤더 등에서 인코더로부터 디코더로 신호될 수 있다.
- [0166] 2.  $M$  및/또는  $N$ 은 표준에 규정된 다른 프로파일(profile)/레벨들/티어(tier)들 내에 다룰 수 있다. CPR/IBC에 대한 크로마 참조 샘플들을 저장하는 다른  $M_c \times c$  화소 버퍼를 사용할 것이 제안된다.
- [0167] a. 하나의 예에서, 4:2:0 비디오에 대해  $M_c = M/2$  및  $N_c = N/2$ 이다.
- [0168] b. 하나의 예에서, 4:2:4 비디오에 대해  $M_c = M$  및  $N_c = N$ 이다.
- [0169] c. 하나의 예에서, 4:2:2 비디오에 대해  $M_c = M$  및  $N_c = N/2$ 이다.
- [0170] d. 이와는 달리  $M_c$  및  $N_c$ 가  $M$  및  $N$ 과 독립적이 될 수 있다.
- [0171] e. 하나의 예에서, 크로마 버퍼는  $C_b$  및  $C_r$ 에 대응하는 2 개의 채널들을 포함한다.
- [0172] f. 하나의 예에서,  $M_c=M$  및  $N_c=N$ 이다.
- [0173] 3. CPR/IBC에 대한 RGB 참조 샘플들을 저장하는 데  $M \times N$  샘플 버퍼를 사용할 것이 제안된다.
- [0174] a. 하나의 예에서, 버퍼 크기는 64x64이다.
- [0175] b. 하나의 예에서, 버퍼 크기는 128x128이다.
- [0176] c. 하나의 예에서, 버퍼 크기는 64x128이다.
- [0177] d. 하나의 예에서, 버퍼 크기는 128x64이다.
- [0178] e. 이와는 달리, 버퍼 크기가 CTU 크기에 대응한다.
- [0179] f. 이와는 달리 버퍼 크기가 가상 파이프라인 데이터 유닛(VPDU) 크기에 대응한다.
- [0180] 4. 버퍼가 루프 필터링(loop-filtering) 이전에 재생 화소(reconstructed pixel)들을 저장할 수 있을 것이 제안된다. 루프 필터링은 블록 해제 필터(deblocking filter), 적응 루프 필터(adaptive loop filter; ALF), 샘플 적응 오프셋(sample adaptive offset; SAO) (필터), 교차 컴포넌트(cross-component) ALF, 또는 어떤 다른 필터들을 지칭할 수 있다.
- [0181] a. 하나의 예에서, 버퍼는 현재 CTU 내의 샘플들을 저장할 수 있다.
- [0182] b. 하나의 예에서, 버퍼는 현재 CTU 외부의 샘플들을 저장할 수 있다.
- [0183] c. 하나의 예에서, 버퍼는 현재 화면의 모든(any) 부분으로부터 샘플들을 저장할 수 있다.

- [0184] d. 하나의 예에서, 버퍼는 다른 화면들로부터 샘플들을 저장할 수 있다.
- [0185] 5. 버퍼가 루프 필터링 이후에 재생 화소들을 저장할 수 있을 것이 제안된다. 루프 필터링은 블록 해제 필터, 적응 루프 필터(ALF), 샘플 적응 오프셋(SAO) (필터), 교차 컴포넌트 ALF, 또는 어떤 다른 필터들을 지칭할 수 있다.
- [0186] a. 하나의 예에서, 버퍼는 현재 CTU 내의 샘플들을 저장할 수 있다.
- [0187] b. 하나의 예에서, 버퍼는 현재 CTU 외부의 샘플들을 저장할 수 있다.
- [0188] c. 하나의 예에서, 버퍼는 현재 화면의 모든 부분으로부터 샘플들을 저장할 수 있다.
- [0189] d. 하나의 예에서, 버퍼는 다른 화면들로부터 샘플들을 저장할 수 있다.
- [0190] 6. 버퍼가 루프 필터링 이전과 루프 필터링 이후 모두에서 재생 화소들을 저장할 수 있을 것이 제안된다. 루프 필터링은 블록 해제 필터, 적응 루프 필터(ALF), 샘플 적응 오프셋(SAO) (필터), 교차 컴포넌트 ALF, 또는 어떤 다른 필터들을 지칭할 수 있다.
- [0191] a. 하나의 예에서, 버퍼는 그 샘플들의 사용 가능성에 따라 현재 화면 및 다른 화면들 양자로부터 샘플들을 저장할 수 있다.
- [0192] b. 하나의 예에서, 다른 화면들로부터의 참조 샘플들은 루프 필터링 이후의 재생 샘플들이다.
- [0193] c. 하나의 예에서, 다른 화면들로부터의 참조 샘플들은 루프 필터링 이전의 재생 샘플들이다.
- [0194] 7. 버퍼가 코딩된 비디오 데이터의 비트심도(bit-depth)와 다를 수 있는 특정(given) 비트심도를 갖는 샘플들을 저장할 것이 제안된다.
- [0195] a. 하나의 예에서, 재생 버퍼/코딩된 비디오 데이터에 대한 비트심도가 버퍼에 저장된 IBC 참조 샘플들의 그것보다 더 크다.
- [0196] b. 하나의 예에서, (10 비트 대 8 비트) 등 내부 비트심도가 비디오 시퀀스의 입력 비트 심도와 다른 경우라도, IBC 참조 샘플들은 입력 비트심도에 정렬되도록 저장된다.
- [0197] c. 하나의 예에서, 비트심도는 재생 버퍼의 그것과 같다.
- [0198] d. 하나의 예에서, 비트심도는 입력 화상/비디오의 그것과 같다.
- [0199] e. 하나의 예에서, 비트심도는 사전 설정된 수와 같다.
- [0200] f. 한 예에서, 비트심도는 표준의 프로파일에 좌우된다.
- [0201] g. 하나의 예에서, 비트심도 또는 출력 비트심도/입력 비트심도/내부 비트심도와 비교한 비트 심도 차이는 SPS/PPS/시퀀스 헤더/화면 헤더/슬라이스 헤더/타일 그룹 헤더/타일 헤더 또는 다른 종류의 비디오 데이터 유닛 내에 신호될 수 있다.
- [0202] h. 제안된 방법은 다른 항목(bullet)들에서 언급된 본 발명(proposed) 버퍼 정의와 함께 적용될 수 있는데, 이와는 달리 IBC의 기존 설계에도 역시 적용될 수 있다.
- [0203] i. 버퍼의 각 칼라 컴포넌트의 비트심도는 다를 수 있다.
- [0204] **버퍼 초기화(Buffer initiation)**
- [0205] 8. 버퍼를 특정 값으로 초기화(initialize)시킬 것이 제안된다.
- [0206] a. 하나의 예에서, 버퍼는 특정 값으로 초기화된다.
- [0207] i. 하나의 예에서, 특정 값은 입력 비트심도 및/또는 내부 비트심도에 좌우될 수 있다.
- [0208] ii. 하나의 예에서, 버퍼가 예를 들어 8 비트 신호에 대해 128 또는 10 비트 신호에 대해 512 등의 미드그레이 값(mid-grey value)으로 초기화된다.
- [0209] iii. 하나의 예에서, ILR이 사용될 때 버퍼는 forwardLUT(m)으로 초기화된다. 예를 들어  $m = 1 \ll (\text{Bitdepth} - 1)$ 이다.

- [0210] b. 이와는 달리, 버퍼가 SPS/VPS/APS/PPS/시퀀스 헤더/타일 그룹 헤더/화면 헤더/타일/CTU/코딩 유닛(Coding unit)/VPDU/영역에서 신호된 값으로 초기화된다.
- [0211] c. 하나의 예에서, 특정 값은 이전에 디코딩된 화면 또는 슬라이스 또는 CTU 행(row) 또는 CTU 또는 CU들의 샘플들로부터 도출될 수 있다.
- [0212] d. 특정 값은 다른 칼라 컴포넌트에 대해 다를 수 있다.
- [0213] 9. 이와는 달리 이전에 코딩된 블록들로부터의 디코딩된 화서들로 버퍼를 초기화시킬 것이 제안된다.
- [0214] a. 하나의 예에서, 디코딩된 화소들은 루프-내 필터링 이전의 것들이다.
- [0215] b. 하나의 예에서, 버퍼 크기가 CTU와 같을 때, 사용 가능하다면 이전에 디코딩된 CTU의 디코딩된 화소들로 버퍼가 초기화된다.
- [0216] c. 하나의 예에서, 버퍼 크기가 64x64일 때, 사용 가능하다면 이전에 디코딩된 64x64 블록의 디코딩된 화소들로 버퍼 크기가 초기화된다.
- [0217] d. 이와는 달리, 뿐만 아니라, 사용 가능한 이전에 코딩된 블록들이 없으면, 항목 8의 방법들이 적용될 수 있다.
- [0218] **버퍼에 대한 참조(Reference to the buffer)**
- [0219] 10. 블록이 버퍼 내의 화소들을 기준으로 사용하기 위해, 어디서 기준을 얻을지 표시하는 위치(x,y),  $x=0,1,2, \dots, M-1; y=0,1,2, \dots, N-1$ 를 사용할 수 있다.
- [0220] 11. 이와는 달리, 참조 위치가  $l = y*M+x$ ,  $l=0,1, \dots, M*N-1$ 로 표기될 수 있다.
- [0221] 12. 현재 CTU에 관련된 블록의 상-좌(upper-left) 위치를 (x0, y0)로 표기하고, 버퍼 내의 어디서 기준을 얻을지 표시하도록 블록 벡터(BVx,BVy)=(x-x0,y-y0)가 디코더로 전송된다.
- [0222] 13. 이와는 달리, Tx 및 Ty가 소정의 오프셋들일 때 블록 벡터(BVx,BVy)가 (x-x0+Tx,y-y0+Ty)로 정의될 수 있다.
- [0223] 14. 어떤 화소(x0, y0) 및 (BVx, BVy)에 대해, 그 버퍼 내의 기준은 (x0+BVx, y0+BVy)에서 찾을 수 있다.
- [0224] a. 하나의 예에서, (x0+BVx, y0+BVy)가 버퍼 외부에 있으면, 이는 경계에서 클리핑될 (clipped) 것이다.
- [0225] b. 이와는 달리, (x0+BVx, y0+BVy)가 버퍼 외부에 있으면, 그 기준 값은 예를 들어 미드 그레이(mid-grey) 등의 특정 값으로 사전 결정된다.
- [0226] c. 이와는 달리 참조 위치가 ((x0+BVx) mod M, (y0+BVy) mod N)로 정의되어 항상 버퍼 내부에 위치한다.
- [0227] 15. 어떤 화소(x0, y0) 및 (BVx, BVy)에 대해 (x0+BVx, y0+BVy)가 버퍼 외부에 있으면, 그 기준 값은 버퍼 내의 값들로부터 도출될 수 있다.
- [0228] a. 하나의 예에서, 이 값은 버퍼 내의 샘플((x0+BVx) mod M, (y0+BVy) mod N)로부터 도출된다.
- [0229] b. 하나의 예에서, 이 값은 버퍼 내의 샘플((x0+BVx) mod M, clip(y0+BVy, 0, N-1))로부터 도출된다.
- [0230] c. 하나의 예에서, 이 값은 버퍼 내의 샘플(clip(x0+BVx, 0, M-1), (y0+BVy) mod N)로부터 도출된다.
- [0231] d. 하나의 예에서, 이 값은 버퍼 내의 샘플(clip(x0+BVx, 0, M-1), clip(y0+BVy, 0, N-1))로부터 도출된다.
- [0232] 16. 버퍼 범위 외부의 어떤 좌표는 허용되지 않을 수 있다.
- [0233] a. 하나의 예에서, CTU의 상-좌 모서리에 관련된 어떤 화소(x0, y0)와 블록 벡터(BVx, BVy)에 대해, y0+BVy가 [0, ...,N-1]의 범위 내에 있어야 한다는 것이 비트스트림 제약이다.

- [0234] b. 하나의 예에서, CTU의 상-좌 모서리에 관련된 어떤 화소( $x_0, y_0$ )와 블록 벡터( $BV_x, BV_y$ )에 대해,  $x_0 + BV_x$ 가  $[0, \dots, M-1]$ 의 범위 내에 있어야 한다는 것이 비트스트림 제약이다.
- [0235] c. 하나의 예에서, CTU의 상-좌 모서리에 관련된 어떤 화소( $x_0, y_0$ )와 블록 벡터( $BV_x, BV_y$ )에 대해,  $y_0 + BV_y$ 가  $[0, \dots, N-1]$ 의 범위 내에 있고  $x_0 + BV_x$ 가  $[0, \dots, M-1]$ 의 범위 내에 있어야 한다는 것 양자가 비트스트림 제약이다.
- [0236] 17. 한 블록의 신호된 또는 도출된 블록 벡터가 버퍼 외부의 어디를 가리키면 버퍼에 따라 패딩(padding)이 적용될 수 있다.
- [0237] a. 하나의 예에서, 버퍼 외부의 어떤 샘플의 값은 소정 값으로 정의된다.
- [0238] i. 하나의 예에서, 이 값은 예를 들어 8비트 신호에 대해 128, 10비트 신호에 대해 512 등  $1 \ll (\text{Bitdepth}-1)$ 이 될 수 있다.
- [0239] ii. 하나의 예에서, ILR이 사용될 때 이 값은 예를 들어  $m = 1 \ll (\text{Bitdepth}-1)$  등  $\text{forwardLUT}(m)$ 이 될 수 있다.
- [0240] iii. 이와는 달리, 소정 값의 표시가 SPS/PPS/시퀀스 헤더/화면 헤더/슬라이스 헤더/타일 그룹/타일/CTU/CU 레벨에서 신호되거나 표시될 수 있다.
- [0241] b. 하나의 예에서, 버퍼 외부의 어떤 샘플은 버퍼 내의 가장 가까운 샘플의 값으로 정의된다.
- [0242] 18. 버퍼 외부의 기준을 취급하는 방법이 수평 및 수직으로 다르거나 (예를 들어 화면 경계에 더 가까운지 아닌지 등) 현재 블록의 위치에 따라 다를 수 있다.
- [0243] a. 하나의 예에서,  $y_0 + BV_y$ 가  $[0, N-1]$ 의 밖에 있으면,  $(x_0 + BV_x, y_0 + BV_y)$ 의 샘플 값은 소정 값으로 할당된다.
- [0244] b. 하나의 예에서  $x_0 + BV_x$ 가  $[0, M-1]$ 의 밖에 있으면,  $(x_0 + BV_x, y_0 + BV_y)$ 의 샘플 값은 소정 값으로 할당된다.
- [0245] c. 이와는 달리,  $(x_0 + BV_x, y_0 + BV_y)$ 의 샘플 값이  $((x_0 + BV_x) \bmod M, y_0 + BV_y)$ 의 샘플 값으로 할당되는데,  $((x_0 + BV_x) \bmod M, y_0 + BV_y)$ 가 여전히 버퍼 밖에 있으면 값을 더 도출하기 위한 다른 방법을 호출(invok e)할 수 있다.
- [0246] d. 이와는 달리,  $(x_0 + BV_x, y_0 + BV_y)$ 의 샘플 값이  $(x_0 + BV_x, (y_0 + BV_y) \bmod N)$ 의 샘플 값으로 할당되는데,  $(x_0 + BV_x, (y_0 + BV_y) \bmod N)$ 가 여전히 버퍼 외부에 있으면 값을 더 도출하기 위한 다른 방법을 호출할 수 있다.
- [0247] **블록 벡터 표현(Block vector representation)**
- [0248] 19. 블록 벡터( $BV_x, BV_y$ )의 각 컴포넌트 또는 컴포넌트 중의 하나는 어떤 범위에 대해 정규화(normalize)된다.
- [0249] a. 하나의 예에서,  $BV_x$ 는  $(BV_x \bmod M)$ 로 대체될 수 있다.
- [0250] b. 이와는 달리,  $X$ 가 소정 값일 때  $BV_x$ 가  $((BV_x + X) \bmod M) - X$ 로 대체될 수 있다.
- [0251] i. 하나의 예에서,  $X$ 는 64이다.
- [0252] ii. 하나의 예에서,  $X$ 는  $M/2$ 이다.
- [0253] iii. 하나의 예에서,  $X$ 는 현재 CTU에 대한 블록의 수평 좌표이다.
- [0254] c. 하나의 예에서,  $BV_y$ 가  $(BV_y \bmod N)$ 로 대체될 수 있다.
- [0255] d. 이와는 달리,  $Y$ 가 소정 값일 때  $BV_y$ 가  $((BV_y + Y) \bmod N) - Y$ 로 대체될 수 있다.
- [0256] i. 하나의 예에서,  $Y$ 는 64이다.
- [0257] ii. 하나의 예에서,  $Y$ 는  $N/2$ 이다.
- [0258] iii. 하나의 예에서,  $Y$ 는 현재 CTU에 대한 블록의 수직 좌표이다.
- [0259] 20.  $BV_x$ 와  $BV_y$ 는 다른 정규화 범위(normalized range)들을 가질 수 있다.

- [0260] 21. 블록 벡터 차이(BVDx, BVDy)는 어떤 범위에 대해 정규화될 수 있다.
- [0261] a. 하나의 예에서, 함수 mod가 나머지(remainder)를 회신할 때 BVDx가  $(BVDx \bmod M)$ 로 대체될 수 있다.
- [0262] b. 이와는 달리, X가 소정 값일 때 BVDx가  $((BVDx+X) \bmod M)-X$ 로 대체될 수 있다.
- [0263] i. 하나의 예에서, X는 64이다.
- [0264] ii. 하나의 예에서, X는  $M/2$ 이다.
- [0265] c. 하나의 예에서, BVDy가  $(BVDy \bmod N)$ 로 대체될 수 있다.
- [0266] d. 이와는 달리, Y가 소정 값일 때 BVDy가  $((BVDy+Y) \bmod N)-Y$ 로 대체될 수 있다.
- [0267] i. 하나의 예에서, Y는 64이다.
- [0268] ii. 하나의 예에서, Y는  $N/2$ 이다;
- [0269] 22. BVDx와 BVDy는 다른 정규화 범위들을 가질 수 있다.
- [0270] **블록 벡터에 대한 유효성 점검(Validity check for a block vector)**
- [0271] IBC 버퍼의 폭과 높이를  $W_{buf}$  및  $H_{buf}$ 로 표기하기로 한다. (루마 블록, 크로마 블록, CU, TU, 4x4, 2x2 또는 다른 서브블록(subblock)들이 될 수 있는)  $W \times H$  블록에 대해, 화면의 상-좌 모서리에 관련된 (X, Y)로부터 초기화하여 블록 벡터(BVx, BVy)가 유효한지 아닌지 구분(tell)하는 데 다음이 적용될 수 있다.  $W_{pic}$  및  $H_{pic}$ 이 화면의 폭 및 높이이고;  $W_{ctu}$  및  $H_{ctu}$ 가 CTU의 폭 및 높이라고 가정한다. 함수 floor(x)는 x보다 크지 않은 최대 정수를 회신한다. 함수 isRec(x, y)는 샘플(x, y)이 재생되었는지를 회신한다.
- [0272] 23. 참조 위치가 화면 경계의 외부에 있더라도 블록 벡터(BVx, BVy)가 유효한 것으로 설정될 수 있다.
- [0273] a. 하나의 예에서,  $X+BVx < 0$ 이더라도 블록 벡터가 유효한 것으로 설정될 수 있다.
- [0274] b. 하나의 예에서,  $X+W+BVx > W_{pic}$ 이더라도 블록 벡터가 유효한 것으로 설정될 수 있다.
- [0275] c. 하나의 예에서,  $Y+BVy < 0$ 이더라도 블록 벡터가 유효한 것으로 설정될 수 있다.
- [0276] d. 하나의 예에서,  $Y+H+BVy > H_{pic}$ 이더라도 블록 벡터가 유효한 것으로 설정될 수 있다.
- [0277] 24. 어떤 참조 위치가 현재 CTU 행 외부에 있더라도 블록 벡터(BVx, BVy)가 유효한 것으로 설정될 수 있다.
- [0278] a. 하나의 예에서,  $Y+BVy < \text{floor}(Y/H_{ctu}) * H_{ctu}$ 이더라도 블록 벡터가 유효한 것으로 설정될 수 있다.
- [0279] b. 하나의 예에서,  $Y+H+BVy \geq \text{floor}(Y/H_{ctu}) * H_{ctu} + H_{ctu}$ 이더라도 블록 벡터가 유효한 것으로 설정될 수 있다.
- [0280] 25. n이 IBC에 대한 기준 영역으로 사용될 수 있는 (현재 CTU를 포함 또는 제외한) CTU들의 수일 때, 어떤 참조 위치가 현재 및 좌측 (n-1)개의 CTU들 외부에 있더라도 블록 벡터(BVx, BVy)가 유효한 것으로 설정될 수 있다.
- [0281] a. 하나의 예에서,  $X+BVx < \text{floor}(X/W_{ctu}) * W_{ctu} - (n-1) * W_{ctu}$ 이더라도 블록 벡터가 유효한 것으로 설정될 수 있다.
- [0282] b. 하나의 예에서,  $X+W+BVx > \text{floor}(X/W_{ctu}) * W_{ctu} + W_{ctu}$ 이더라도 블록 벡터가 유효한 것으로 설정될 수 있다.
- [0283] 26. 어떤 샘플이 재생되지 않았더라도 블록 벡터(BVx, BVy)가 유효한 것으로 설정될 수 있다.
- [0284] a. 하나의 예에서, isRec( $X+BVx$ ,  $Y+BVy$ )가 거짓이더라도 블록 벡터가 유효한 것으로 설정될 수 있다.
- [0285] b. 하나의 예에서, isRec( $X+BVx+W-1$ ,  $Y+BVy$ )가 거짓이더라도 블록 벡터가 유효한 것으로 설정

될 수 있다.

[0286] c. 하나의 예에서,  $\text{isRec}(X+Bv_x, Y+Bv_y + H-1)$ 이 거짓이더라도 블록 벡터가 유효한 것으로 설정될 수 있다.

[0287] d. 하나의 예에서,  $\text{isRec}(X+Bv_x + W-1, Y+Bv_y + H-1)$ 이 거짓이더라도 블록 벡터가 유효한 것으로 설정될 수 있다.

[0288] 27. 블록이 CTU 행의 첫 번째 CTU의 것이 아니면 블록 벡터( $Bv_x, Bv_y$ )가 항상 유효한 것으로 설정될 수 있다.

[0289] a. 이와는 달리, 블록 벡터가 항상 유효한 것으로 설정될 수 있다.

[0290] 28. 블록 벡터( $Bv_x, Bv_y$ )는 다음 3가지 조건들이 모두 충족되면 항상 유효한 것으로 설정될 수 있다.

[0291] -  $X + Bv_x \geq 0$

[0292] -  $Y + Bv_y \geq \text{floor}(Y/H_{\text{ctu}})$

[0293] -  $\text{isRec}(X + Bv_x + W - 1, Y + Bv_y + H - 1) == \text{true}$

[0294] a. 이와는 달리, CTU 행의 첫 번째 CTU의 블록에 대해 이 3가지 조건들이 충족되면, 블록 벡터가 항상 유효한 것으로 설정될 수 있다.

[0295] 29. 블록 벡터( $Bv_x, Bv_y$ )가 유효하면, 블록에 대한 샘플 복사(sample copying)는 블록 벡터에 기반할 수 있다.

[0296] a. 하나의 예에서, 샘플( $X, Y$ )의 예측은  $((X+Bv_x)\%W_{\text{buf}}, (Y+Bv_y)\%H_{\text{buf}})$ 로부터 이뤄질 수 있다.

#### [0297] 버퍼 갱신(Buffer update)

[0298] 30. 새로운 화면 또는 타일을 코딩할 때 버퍼가 리셋될 수 있다.

[0299] a. "리셋(reset)"이라는 용어는 버퍼가 초기화(initialize)되는 것을 지칭할 수 있다.

[0300] b. "리셋(reset)"이라는 용어는 버퍼 내의 모든 샘플들/화소들이 (예를 들어 0 또는 -1의) 특정 값으로 설정되는 것을 지칭할 수 있다.

[0301] 31. VPDU의 코딩을 완료하면 버퍼는 VPDU의 재생 값(reconstructed value)들로 갱신될 수 있다.

[0302] 32. CTU의 코딩을 완료하면 버퍼는 CTU의 재생 값들로 갱신될 수 있다.

[0303] a. 하나의 예에서, 버퍼가 가득차지(full) 않으면 버퍼는 CTU별로(CTU by CTU) 순차적으로 갱신될 수 있다.

[0304] b. 하나의 예에서, 버퍼가 가득차면, 가장 오래된(oldest) CTU에 대응하는 버퍼 영역이 갱신될 것이다.

[0305] c. 하나의 예에서,  $M=mW$  및  $N=H$  ( $W$  및  $H$ 는 CTU 크기이고;  $M$  및  $N$ 은 버퍼 크기)이고 이전 갱신 영역이  $(kW, 0)$ 로부터 시작하면, 갱신될 다음 시작 위치는  $((k+1)W \bmod M, 0)$ 이 될 것이다.

[0306] 33. 버퍼는 각 CTU 행의 시작(beginning)시 리셋될 수 있다.

[0307] a. 이와는 달리, 버퍼는 각 CTU의 디코딩의 시작시 리셋될 수 있다.

[0308] b. 이와는 달리, 버퍼는 하나의 타일의 디코딩의 시작시 리셋될 수 있다.

[0309] c. 이와는 달리, 버퍼는 하나의 타일 그룹/화면의 디코딩의 시작시 리셋될 수 있다.

[0310] 34.  $(x, y)$ 로 시작되는 블록의 코딩이 완료되면,  $(x, y)$ 에서 시작되는 버퍼의 대응 영역이 블록으로부터의 재생으로 갱신될 것이다.

[0311] a. 하나의 예에서,  $(x, y)$ 는 CTU의 상-좌 모서리에 관련된 위치이다.

[0312] 35. 화면에 대한 하나의 블록의 코딩을 완료하면 버퍼의 대응 영역이 블록으로부터의 재생으로 갱신될 것이다.

[0313] a. 하나의 예에서, 버퍼의 위치  $(x \bmod M, y \bmod N)$ 에서의 값이 화면의 상-좌 모서리에 관련된 위치  $(x, y)$ 의 재생 화소값으로 갱신될 수 있다.

- [0314] b. 하나의 예에서, 버퍼의 위치( $x \bmod M$ ,  $y \bmod N$ )에서의 값이 현재 타일의 상-좌 모서리에 관련된 위치( $x$ ,  $y$ )의 재생 화소값으로 갱신될 수 있다.
- [0315] c. 하나의 예에서, 버퍼의 위치( $x \bmod M$ ,  $y \bmod N$ )에서의 값이 현재 CTU 행의 상-좌 모서리에 관련된 위치( $x$ ,  $y$ )의 재생 화소값으로 갱신될 수 있다.
- [0316] d. 하나의 예에서, 버퍼 내의 값이 비트심도 정렬 이후의 재생 화소값들로 갱신될 수 있다.
- [0317] 36. ( $x_b$ ,  $y_b$ )와 ( $x$ ,  $y$ )가 2개의 다른 좌표들일 때, ( $x$ ,  $y$ )로부터 시작되는 코딩이 완료되면, ( $x_b$ ,  $y_b$ )로부터 시작되는 버퍼의 해당 영역이 블록으로부터의 재생으로 갱신될 것이다.
- [0318] a. 하나의 예에서, ( $x$ ,  $y$ )는 CTU의 상-좌 모서리에 관련된 위치이고,  $update\_x$  및  $update\_y$ 가 버퍼 내의 갱신 위치를 가리킬 때 ( $x_b$ ,  $y_b$ )는 ( $x+update\_x$ ,  $y+update\_y$ )이다.
- [0319] 37. 이상의 예들에 있어서, 블록의 재생 값들은 (예를 들어 블록 해제 필터(deblocking filter) 등의) 필터가 적용되기 이전의 재생 값들을 나타낼(indicate) 수 있다.
- [0320] a. 이와는 달리, 블록의 재생 값들은 (예를 들어 블록 해제 필터 등의) 필터가 적용되기 이후의 재생 값들을 나타낼 수 있다.
- [0321] 38. 버퍼가 재생 샘플들로부터 갱신될 때, 재생 샘플들은 샘플 비트심도가 변경되는 등 저장 전에 먼저 수정될 수 있다.
- [0322] a. 하나의 예에서, 버퍼는 버퍼의 bitdepth에 대한 비트심도 정렬 이후의 재생 샘플 값으로 갱신된다.
- [0323] b. 하나의 예에서,  $p$ 가 재생 샘플 값이고  $b$ 가 소정의 비트시프트(bit-shifting) 값일 때, 버퍼 값이 값  $\{p+[1 \ll (b-1)]\} \gg b$ 에 따라 갱신된다.
- [0324] c. 하나의 예에서,  $p$ 가 재생 샘플 값이고  $b$ 가 소정의 비트시프트 값이며 비트심도가 버퍼 비트심도일 때 버퍼 값이 값  $\text{clip}(\{p+[1 \ll (b-1)]\} \gg b, 0, (1 \ll \text{bitdepth})-1)$ 에 따라 갱신된다.
- [0325] d. 하나의 예에서,  $p$ 가 재생 샘플 값이고  $b$ 가 소정의 비트시프트 값일 때, 버퍼 값이 값  $\{p+[1 \ll (b-1)-1]\} \gg b$ 에 따라 갱신된다.
- [0326] e. 하나의 예에서,  $p$ 가 재생 샘플 값이고  $b$ 가 소정의 비트시프트 값이며 비트심도가 버퍼 비트심도일 때 버퍼 값이 값  $\text{clip}(\{p+[1 \ll (b-1)-1]\} \gg b, 0, (1 \ll \text{bitdepth})-1)$ 에 따라 갱신된다.
- [0327] f. 하나의 예에서, 버퍼 값이 값  $p \gg b$ 에 따라 갱신된다.
- [0328] g. 하나의 예에서, bitdepth가 버퍼 비트심도일 때, 버퍼 값이 값  $\text{clip}(p \gg b, 0, (1 \ll \text{bitdepth})-1)$ 에 따라 갱신된다.
- [0329] h. 이상의 예들에서,  $b$ 는 재생 비트심도 빼기 입력 샘플 비트심도가 될 수 있다.
- [0330] 39. 예측을 형성하기 위해 버퍼 샘플들을 사용할 때 전처리(preprocessing)가 적용될 수 있다.
- [0331] a. 하나의 예에서,  $p$ 가 버퍼 내의 샘플 값이고  $b$ 가 소정 값일 때 예측 값은  $p \ll b$ 이다.
- [0332] b. 하나의 예에서, bitdepth가 재생 샘플들의 비트심도일 때, 예측 값은  $\text{clip}(p \ll b, 0, 1 \ll \text{bitdepth})$ 이다.
- [0333] c. 하나의 예에서,  $p$ 가 버퍼 내의 샘플 값이고  $b$ 가 소정 값이며 bitdepth가 재생 샘플들의 비트심도일 때, 예측 값은  $(p \ll b) + (1 \ll (\text{bitdepth}-1))$ 이다.
- [0334] d. 이상의 예들에서,  $b$ 는 재생 비트심도 빼기 입력 샘플 비트심도가 될 수 있다.
- [0335] 40. 버퍼는 특정 순서로 갱신될 수 있다.
- [0336] a. 하나의 예에서, 버퍼는 순차적으로 갱신될 수 있다.
- [0337] b. 하나의 예에서, 버퍼는 재생된 블록들의 순서로 갱신될 수 있다.
- [0338] 41. 버퍼가 가득차면, 버퍼 내의 샘플들은 최근의 재생 샘플들로 대체될 수 있다.

- [0339] a. 하나의 예에서, 샘플들은 선입선출(first-in-first-out) 방식으로 갱신될 수 있다.
- [0340] b. 하나의 예에서, 가장 오래된 샘플들이 교체될 것이다.
- [0341] c. 하나의 예에서, 샘플들에 우선권(priority)이 할당되어 우선권에 따라 교체될 수 있다.
- [0342] d. 하나의 예에서, 샘플들이 "장기(long-term)"로 마킹될 수 있어 다른 샘플들이 먼저 교체될 것이다.
- [0343] e. 하나의 예에서, 높은 우선권을 표시하기 위한 플래그(flag)이 블록과 함께 전송될 수 있다.
- [0344] f. 하나의 예에서, 우선권을 표시하기 위해 숫자가 블록과 함께 전송될 수 있다.
- [0345] g. 하나의 예에서, 어떤 특징을 갖는 재생 블록으로부터의 샘플에 더 높은 우선권이 할당되어 다른 샘플들이 먼저 교체될 것이다.
- [0346] i. 하나의 예에서, IBC 모드에서 코딩된 샘플들의 비율(percentage)이 임계값보다 더 크면, 블록의 모든 샘플들에 높은 우선권이 할당될 수 있다.
- [0347] ii. 하나의 예에서, 팔레트(pallet) 모드에서 코딩된 샘플들의 비율이 임계값보다 더 크면, 블록의 모든 샘플들에 높은 우선권이 할당될 수 있다.
- [0348] iii. 하나의 예에서, IBC 또는 팔레트 모드에서 코딩된 샘플들의 비율이 임계값보다 더 크면, 블록의 모든 샘플들에 높은 우선권이 할당될 수 있다.
- [0349] iv. 하나의 예에서, 변환 스킵(transform-skip) 모드에서 코딩된 샘플들의 비율이 임계값보다 더 크면, 블록의 모든 샘플들에 높은 우선권이 할당될 수 있다.
- [0350] v. 임계값은 블록 크기, 칼라 컴포넌트, CTU 크기에 따라 다를 수 있다.
- [0351] vi. 임계값은 SPS/PPS/시퀀스 헤더/슬라이스 헤더/타일 그룹/타일 레벨/영역 내에 실행될 수 있다.
- [0352] h. 하나의 예에서, 버퍼가 가득 찼다는 것은 버퍼 내의 사용 가능한 샘플들의 수가 특정 임계값과 같거나 더 크음을 의미한다.
- [0353] i. 하나의 예에서, 버퍼 내의 사용 가능한 샘플들의 수가 64x64x3 루마 샘플들과 같거나 더 크면, 이 버퍼는 가득 찼다고 판단된다.
- [0354] **대체적인 버퍼 조합(Alternative buffer combination)**
- [0355] 42. 이전에 코딩된 3개의 64x64 블록들을 기준 영역으로 항상 사용하는 대신, 현재 블록(또는 VPDU)의 위치에 기반하여 이를 적응적으로(adaptively) 변경할 것이 제안된다.
- [0356] a. 하나의 예에서, 64x4 블록을 코딩/디코딩할 때, 이전 3개의 64x4 블록들이 기준으로 사용될 수 있다. 도 2와 비교하여, 이전 64x4 블록들의 더 많은 종류의 조합들이 적용될 수 있다. 도 2는 이전 64x4 블록들의 다른 조합의 예를 보인다.
- [0357] 43. z-스캔 순서를 사용하는 대신, 수직 스캔 순서가 사용될 수 있다.
- [0358] a. 하나의 예에서, 하나의 블록이 z-스캔 순서로 인덱스 0..3을 갖는 4개의 VPDU들로 분할되면 인코딩/디코딩 순서는 0, 2, 1, 3이다.
- [0359] b. 하나의 예에서, 64x4 블록들을 인코딩/디코딩할 때, 이전 3개의 64x4 블록들이 기준으로 사용될 수 있다. 도 2에 비교하여, 64x4 블록들의 더 많은 종류의 코딩/디코딩 순서들이 적용될 수 있다. 도 4는 64x4 블록들의 다른 코딩/디코딩 순서들의 예를 보인다.
- [0360] c. 이와는 달리, 이상의 방법들이 스크린 콘텐츠 코딩(screen content coding)에만 적용될 수 있다.
- [0361] d. 이와는 달리, 이상의 방법들이, CPR이 하나의 타일/타일 그룹/화면에 대해 이네이블되었을 때만 적용될 수 있다.
- [0362] e. 이와는 달리, 이상의 방법들이, CPR이 하나의 CTU 또는 하나의 CTU 행에 대해 이네이블되었

을에만 적용될 수 있다

[0363] 가상 IBC 버퍼(Virtual IBC buffer)

[0364] 이하에서 VPDU의 폭과 높이는 루마 샘플들에서 각각  $W_{VPDU}$  (예를 들어 64)과  $H_{VPDU}$  (예를 들어 64)로 표기된다. 이와는 달리,  $W_{VPDU}$  및/또는  $H_{VPDU}$ 가 (예를 들어 CTU 등) 다른 비디오 유닛의 폭 및/또는 높이를 표기할 수 있다.

[0365] 44. 가상 버퍼(virtual buffer)가 IBC 기준 영역 상태를 지속적으로 추적(keep track)하도록 유지될 수 있다.

[0366] a. 하나의 예에서, 가상 버퍼 크기가  $mW_{VPDU} \times nH_{VPDU}$ 이다.

[0367] i. 하나의 예에서, m은 3이고 n은 2이다.

[0368] ii. 하나의 예에서, m 및/또는 n은 화면 해상도와 CTU 크기들에 좌우될 수 있다.

[0369] iii. 하나의 예에서, m 및/또는 n은 신호되거나 사전에 규정(pre-defined)될 수 있다.

[0370] b. 하나의 예에서, 이상의 항목(bullet)들과 하부 항목(sub-bullet)에 기재된 방법들이 가상 버퍼에 적용될 수 있다.

[0371] c. 하나의 예에서, 화면/슬라이스/타일/브릭의 상-좌 모서리에 대한 샘플(x, y)이  $(x\%(mW_{VPDU}), y\%(nH_{VPDU}))$ 로 매핑될 수 있다.

[0372] 45. 배열(array)이 가상 버퍼에 연계된 각 샘플의 사용 가능성(availability)을 추적하는 데 사용될 수 있다.

[0373] a. 하나의 예에서, 플랙이 가상 버퍼 내의 샘플에 연계되어 버퍼 내의 샘플이 IBC 기준으로 사용될 수 있는지 여부를 규정할 수 있다.

[0374] b. 하나의 예에서, 루마 및 크로마 샘플들을 포함하는 각 4x4 블록이 플랙을 공유하여(share) 그 블록에 연계된 어떤 샘플들이 IBC 기준으로 사용될 수 있는지 여부를 표시할 수 있다.

[0375] c. 하나의 예에서, IBC 참조 샘플들의 가용성을 추적하기 위해 유지되는 3 x 2 VPDU(예를 들어, 4x4 블록이 동일한 가용성 플래그를 공유할 수 있다)에 대응하는 어레이.

[0376] d. 하나의 예에서, IBC 참조 샘플들의 가용성을 추적하기 위해 유지되는 4 x 2 VPDU(예를 들어, 4x4 블록이 동일한 가용성 플래그를 공유할 수 있다)에 대응하는 어레이.

[0377] 46. VPDU 또는 비디오 유닛의 디코딩을 완료한 다음, 가상 버퍼에 연계된 어떤 샘플들이 IBC 기준으로 사용 불가능하다고 마킹될 수 있다.

[0378] a. 하나의 예에서, 어느 샘플들이 사용 불가능한 것으로 마킹될 것인지가 가장 최근에 디코딩된 VPDU의 위치에 좌우된다.

[0379] b. 어떤 샘플이 사용 불가능한 것으로 마킹되면, 이 샘플로부터의 예측이 허용되지 않는다.

[0380] i. 이와는 달리, (예를 들어 디폴트값을 사용하는 등의) 다른 방법이 더 적용되어 사용 불가능한 샘플을 대체하는 예측 변수(predictor)를 도출할 수 있다.

[0381] 47. 가장 최근에 디코딩된 VPDU의 위치가, 가상 버퍼에 연계된 어느 샘플들이 사용 불가능한 것으로 마킹될 수 있는지의 식별을 돕도록 기록될 수 있다.

[0382] a. 하나의 예에서, VPDU의 디코딩의 시작시 가장 최근에 디코딩된 VPDU의 위치에 따라 가상 버퍼에 연계된 어떤 샘플들이 사용 불가능한 것으로 마킹될 수 있다.

[0383] i. 하나의 예에서,  $(x_{PrevVPDU}, y_{PrevVPDU})$ 를 가장 최근에 디코딩된 VPDU의 화면/슬라이스/타일/브릭/다른 비디오 처리 유닛의 상-좌 모서리에 대한 상-좌 위치로 표기할 때,  $y_{PrevVPDU}\%(nH_{VPDU})$ 가 0이면, 어떤 위치(x, y)가 사용 불가능한 것으로 마킹될 수 있다.

- [0384] 1. 하나의 예에서,  $x$ 가  $[xPrevVPDU - 2W_{VPDU} + 2mW_{VPDU}) \% mW_{VPDU}, ((xPrevVPDU - 2W_{VPDU} + 2mW_{VPDU}) \% mW_{VPDU}) - 1 + W_{VPDU}]$  등의 범위 이내일 수 있고;
- [0385] 2. 하나의 예에서,  $y$ 가  $[yPrevVPDU \% (nH_{VPDU}), (yPrevVPDU \% (nH_{VPDU})) - 1 + H_{VPDU}]$  등의 범위 이내일 수 있으며;
- [0386] 3. 하나의 예에서,  $x$ 가  $[xPrevVPDU - 2W_{VPDU} + 2mW_{VPDU}) \% mW_{VPDU}, ((xPrevVPDU - 2W_{VPDU} + 2mW_{VPDU}) \% mW_{VPDU}) - 1 + W_{VPDU}]$  등의 범위 이내일 수 있고  $y$ 가  $[yPrevVPDU \% (nH_{VPDU}), (yPrevVPDU \% (nH_{VPDU})) - 1 + H_{VPDU}]$ 의 범위 이내일 수 있다.
- [0387] ii. 하나의 예에서,  $(xPrevVPDU, yPrevVPDU)$ 를 가장 최근에 디코딩된 VPDU의 화면/슬라이스/타일/브릭/다른 비디오 처리 유닛의 상-좌 모서리에 대한 상-좌 위치로 표기할 때,  $yPrevVPDU \% (nH_{VPDU})$ 가 0이 아니면, 어떤 위치( $x, y$ )가 사용 불가능한 것으로 마킹될 수 있다.
- [0388] 1. 하나의 예에서,  $x$ 가  $[xPrevVPDU - W_{VPDU} + 2mW_{VPDU}) \% mW_{VPDU}, ((xPrevVPDU - W_{VPDU} + 2mW_{VPDU}) \% mW_{VPDU}) - 1 + W_{VPDU}]$  등의 범위 이내 일 수 있고;
- [0389] 2. 하나의 예에서,  $y$ 가  $[yPrevVPDU \% (nH_{VPDU}), (yPrevVPDU \% (nH_{VPDU})) - 1 + H_{VPDU}]$  등의 범위 이내 일 수 있으며;
- [0390] 3. 하나의 예에서,  $x$ 가  $[xPrevVPDU - W_{VPDU} + 2mW_{VPDU}) \% mW_{VPDU}, ((xPrevVPDU - W_{VPDU} + 2mW_{VPDU}) \% mW_{VPDU}) - 1 + W_{VPDU}]$ 의 범위 이내가 될 수 있고,  $y$ 가  $[yPrevVPDU \% (nH_{VPDU}), (yPrevVPDU \% (nH_{VPDU})) - 1 + H_{VPDU}]$  등의 범위 이내가 될 수 있다.
- [0391] ,  $((xPrevVPDU - W_{VPDU} + 2mW_{VPDU}) \% mW_{VPDU}) - 1 + W_{VPDU}]$ 의 범위 이내가 될 수 있고,  $y$ 가  $[yPrevVPDU \% (nH_{VPDU}), (yPrevVPDU \% (nH_{VPDU})) - 1 + H_{VPDU}]$  등의 범위 이내가 될 수 있다.
- [0392] 48. CU가 복수의 VPDU들을 포함하면, IBC 기준 사용 가능성 마킹 프로세스를 적용하는 대신, IBC 기준 사용 가능성 마킹 프로세스가 CU에 따를 수 있다.
- [0393] a. 하나의 예에서, 복수의 VPDU들을 포함하는 CU의 디코딩 시작시, CU 내의 VPDU를 디코딩하기 전에 IBC 기준 사용 가능성 마킹 프로세스가 각 VPDU에 대해 적용될 수 있다.
- [0394] b. 이러한 경우, 128x64 및 64x128 IBC 블록들은 허용되지 않을 수 있다.
- [0395] i. 하나의 예에서, 128x64 및 64x128 CU들에 대한 pred\_mode\_ibc\_flag은 전송되지 않고 0이라고 추론될 수 있다.
- [0396] 49. 기준 블록 또는 서브블록(sub-block)에 대해, 기준 블록에 연계된 블록 벡터가 유효한지 여부를 구별(tell)하기 위해 상-우(upper-right) 모서리의 기준 사용 가능성 상태를 점검할 필요가 없을 수 있다.
- [0397] a. 하나의 예에서, 블록 벡터가 유효한지 여부를 구분하기 위해 블록/서브블록의 상-좌(upper-left), 하-좌(bottom-left), 및 하-우(bottom-right) 모서리만 점검될 것이다.
- [0398] 50. IBC 버퍼 크기는 VPDU 크기(폭/높이가 vSize로 표기)에 좌우되거나 및/또는 CTB/CTU 크기(폭/높이가 ctbSize로 표기)에 좌우될 수 있다.
- [0399] a. 하나의 예에서, 버퍼의 높이는 ctbSize와 동일할 수 있다.

- [0400] b. 하나의 예에서, 버퍼의 폭은  $\min(\text{ctbSize}, 64)$ 에 좌우될 수 있다.
- [0401] i. 하나의 예에서, 버퍼의 폭은  $(128*128/\text{vSize}, \min(\text{ctbSize}, 64))$ 가 될 수 있다.
- [0402] 51. IBC버퍼는 화소 범위 외부의 값을 포함할 수 있는데, 이는 이 위치가 예를 들어 다른 샘플들의 예측에 사용되지 않는 등 IBC 기준에 사용 불가능하다는 것을 나타낸다.
- [0403] a. 샘플 값은 이 샘플이 사용 불가능하다는 것을 표시하는 값으로 설정될 수 있다.
- [0404] b. 하나의 예에서, 이 값은 -1이다.
- [0405] c. 하나의 예에서, 이 값은  $[0, 1 \ll (\text{internal\_bit\_depth}) - 1]$  외부의 어떤 값이 될 수 있는데, 여기서  $\text{internal\_bit\_depth}$ 는 양의 정수 값이다. 예를 들어,  $\text{internal\_bit\_depth}$ 는 칼라 컴포넌트에 대한 샘플의 인코딩/디코딩에 사용된 내부 비트심도이다.
- [0406] d. 하나의 예에서, 이 값은  $[0, 1 \ll (\text{input\_bit\_depth}) - 1]$  외부의 어떤 값일 수 있는데, 여기서  $\text{input\_bit\_depth}$ 는 양의 정수 값이다. 예를 들어,  $\text{input\_bit\_depth}$ 는 칼라 컴포넌트에 대한 샘플의 인코딩/디코딩에 사용된 입력 비트심도이다.
- [0407] 52. IBC버퍼 내의 샘플들에 대한 사용 가능성 마킹은 현재 블록의 위치, 현재 블록의 크기, CTU/CTB 크기, 및 VPDU 크기에 좌우될 수 있다. 하나의 예에서,  $(\text{xCb}, \text{yCb})$ 가 화면의 상-좌(top-left)에 대한 블록의 위치를 표기하고;  $\text{ctbSize}$ 가 CTU/CTB의 크기(즉 폭 및/또는 높이)이며;  $\text{vSize} = \min(\text{ctbSize}, 64)$ 이고;  $\text{wIbcBuf}$  및  $\text{hIbcBuf}$  IBC 버퍼의 폭 및 높이라고 한다.
- [0408] a. 하나의 예에서,  $(\text{xCb}\% \text{vSize})$ 가 0이고  $(\text{yCb}\% \text{vSize})$ 가 0이면, IBC 버퍼 내의 위치들의 어떤 세트(set)들은 사용 불가능한 것으로 마킹될 수 있다.
- [0409] b. 하나의 예에서, 현재 블록 크기가 VPDU 크기, 즉  $\min(\text{ctbSize}, 64)$ 보다 더 작으면, 사용 불가능한 것으로 마킹되는 영역은 VPDU 크기에 따를 수 있다.
- [0410] c. 하나의 예에서, 현재 블록 크기가 VPDU 크기, 즉  $\min(\text{ctbSize}, 64)$ 보다 더 크면, 사용 불가능한 것으로 마킹되는 영역은 CU 크기에 따를 수 있다.
- [0411] 53. 화면의 상-좌 위치에 대한 (예를 들어  $\text{VPDU}(\text{xV}, \text{yV})$  등의) 비디오 유닛의 디코딩의 시작시, IBC 버퍼 내의 해당 위치들은 화소 범위 외부의 값으로 설정될 수 있다.
- [0412] a. 하나의 예에서,  $x = \text{xV}, \dots, \text{xV} + \text{ctbSize} - 1$ 이고  $y = \text{yV}, \dots, \text{yV} + \text{ctbSize} - 1$ 일 때 버퍼 내에 위치  $(\text{x}\% \text{wIbcBuf}, \text{y}\% \text{hIbcBuf})$ 를 가지는 버퍼 샘플들은 값 -1로 설정될 것인데, 여기서  $\text{wIbcBuf}$  및  $\text{hIbcBuf}$ 는 IBC 버퍼의 폭 및 높이이고,  $\text{ctbSize}$ 는 CTU/CTB의 폭이다.
- [0413] i. 하나의 예에서,  $\text{hIbcBuf}$ 는  $\text{ctbSize}$ 와 동일할 수 있다.
- [0414] 54. 비트스트림 적합성 제약(bitstream conformance constraint)은 IBC 버퍼 내의 샘플의 값에 따를 수 있다.
- [0415] a. 하나의 예에서, IBC 블록 내의 블록 벡터와 연계된 기준 블록이 화소 범위 외부의 값을 포함하면, 그 비트스트림은 부적합(illegal)하다.
- [0416] 55. 비트스트림 적합성 제약은 IBC 버퍼 내의 사용 가능성 표시(availability indication)에 따를 수 있다.
- [0417] a. 하나의 예에서, IBC 버퍼 내에 매핑된 어떤 참조 샘플이 블록의 인코딩/디코딩에 사용 불가능한 것으로 마킹되면, 그 비트스트림은 부적합하다.
- [0418] b. 하나의 예에서, 단일 트리(singletree)가 사용될 때, 블록의 인코딩/디코딩을 위해 IBC 버퍼 내에 매핑된 어떤 루마 참조 샘플들이 사용 불가능하다고 마킹되면, 그 비트스트림은 부적합하다.
- [0419] c. 적합한 비트스트림(conformance bitstream)은 IBC 코딩된 블록에 대해 연계된 블록 벡터가 IBC 버퍼에 매핑된 기준 블록을 가리키고(point to) 블록의 인코딩/디코딩을 위한 IBC 버퍼 내에 위치하는 각 루마 참조 샘플이 사용가능한 것으로 마킹되어야 한다는 조건을 충족할 수 있다(예를 들어 샘플들의 값들이  $[K0, K1]$ 의 범위 내에 있는데, 여기서 예를 들어  $K0$ 은 0으로 설정되고  $\text{BitDepth}$ 가 내부 비트심도 또는 입력 비트심도일 때  $K1$ 이  $(1 \ll \text{BitDepth} - 1)$ 로 설정된다).
- [0420] 56. 비트스트림 적합성 제약은 분할 트리(partitioning tree) 종류와 현재 CU의 코딩  $\text{treeType}$ 에 좌우될 수 있

다.

- [0421] a. 하나의 예에서, 이중 트리(dualtree)가 (예를 들어 슬라이스/화면/브릭/타일 등) 하이 레벨 (high-level)에서 허용되고 (예를 들어 CU/PU/CB/PB 등의) 현재 비디오 블록이 단일 트리로 코딩되면, 비트스트림 제약은 IBC버퍼 내에 매핑된 모든 컴포넌트의 위치가 사용 불가능한 것으로 마킹되었는지 여부를 점검할 필요가 있을 것이다.
- [0422] b. 하나의 예에서, 이중 트리가 (예를 들어 슬라이스/화면/브릭/타일 등) 하이 레벨에서 허용되고 (예를 들어 CU/PU/CB/PB 등의) 현재 루마 비디오 블록이 이중 트리로 코딩되면, 비트스트림 제약은 IBC 버퍼 내에 매핑된 크로마 컴포넌트의 위치가 사용 불가능한 것으로 마킹되었는지 여부를 무시할 수 있다.
- [0423] i. 이와는 달리, 이러한 경우 비트스트림 제약이 여전히 IBC 버퍼 내에 매핑된 크로마 컴포넌트의 위치가 사용 불가능한 것으로 마킹되었는지 여부를 점검할 수 있다.
- [0424] c. 하나의 예에서, 단일 트리가 사용되면, 비트스트림 제약이 IBC 버퍼 내에 매핑된 크로마 컴포넌트의 위치가 사용 불가능한 것으로 마킹되었는지 여부를 무시할 수 있다.
- [0425] **현재 VTM 설계에 대한 개선(Improvement to the current VTM design)**
- [0426] 57. IBC에 대한 예측(prediction)이 재생(reconstruction)보다 더 낮은 정밀도(precision)를 가질 수 있다.
- [0427] a. 하나의 예에서, 예측 값은  $\text{clip}\{p + \lfloor 1 \ll (b-1) \rfloor\} \gg b, 0, (1 \ll \text{bitdepth}) - 1\} \ll b$ 를 따르는데, 여기서  $p$ 는 재생된 샘플 값이고,  $b$ 는 소정의 비트 시프트(bit-shifting) 값이며,  $\text{bitdepth}$ 는 예측 샘플 비트심도이다.
- [0428] b. 하나의 예에서, 예측 값은  $\text{clip}\{p + \lfloor 1 \ll (b-1) - 1 \rfloor\} \gg b, 0, (1 \ll \text{bitdepth}) - 1\} \ll b$ 를 따르는데, 여기서  $p$ 는 재생 샘플 값,  $b$ 는 소정의 비트 시프트 값이다.
- [0429] c. 하나의 예에서, 예측 값은  $((p \gg b) + (1 \ll (\text{bitdepth} - 1))) \ll b$ 를 따르는데, 여기서  $\text{bitdepth}$ 는 예측 샘플의 비트심도이다.
- [0430] d. 하나의 예에서, 예측 값은  $(\text{clip}((p \gg b), 0, (1 \ll (\text{bitdepth} - b)))) + (1 \ll (\text{bitdepth} - 1))) \ll b$ 를 따르는데, 여기서  $\text{bitdepth}$ 는 예측 샘플의 비트심도이다.
- [0431] e. 하나의 예에서, 예측 값은 ILR이 적용되는지 여부에 따라 다른 방식으로 클리핑(clipped)될 수 있다.
- [0432] f. 이상의 예들에서,  $b$ 는 재생 비트심도 빼기 입력 샘플 비트심도가 될 수 있다.
- [0433] g. 하나의 예에서, 비트심도, 또는 출력 비트심도/입력 비트심도/내부 비트심도와 비교한 비트심도 차이는 SPS/PPS/시퀀스 헤더/화면 헤더/슬라이스 헤더/타일 그룹 헤더/타일 헤더 또는 다른 종류의 비디오 데이터 유닛에 신호될 수 있다.
- [0434] 58. IBC의 예측의 일부는 (재생보다) 더 낮은 정밀도를 갖고 다른 일부는 재생과 동일한 정밀도를 갖는다.
- [0435] a. 하나의 예에서, 허용된 기준 영역은 다른 정밀도(예를 들어 비트심도)들을 갖는 샘플들을 포함할 수 있다.
- [0436] b. 하나의 예에서, 다른 64x64 블록들로부터의 기준이 디코딩되는 현재 64x64블록보다 낮은 정밀도를 갖고 현재 64x64 블록은 재생과 동일한 정밀도를 갖는다.
- [0437] c. 하나의 예에서, 다른 CTU들로부터의 기준이 디코딩되는 현재 CTU보다 낮은 정밀도를 갖고 현재 CTU로부터의 기준이 재생과 동일한 정밀도를 갖는다.
- [0438] d. 하나의 예에서, 어떤 칼라 컴포넌트들로부터의 기준이 낮은 정밀도를 갖고 다른 칼라 컴포넌트로부터의 기준이 재생과 동일한 정밀도를 갖는다.
- [0439] 59. CTU 크기가  $M \times M$ 이고 기준 영역 크기가  $nM \times nM$ 이면, 기준 영역은 CTU 행에서 가장 가까운 사용 가능한  $n \times n$  CTU들이다.
- [0440] a. 하나의 예에서, 기준 영역 크기가  $128 \times 128$ 이고 CTU 크기가  $64 \times 64$ 이면, CTU 행에서 가장 가까운 사용 가능한 4개의 CTU들이 IBC 기준에 사용될 수 있다.

- [0441] b. 하나의 예에서, 기준 영역 크기가  $128 \times 128$ 이고 CTU 크기가  $32 \times 32$ 이면, CTU 행에서 가장 가까운 사용 가능한 16개의 CTU들이 IBC 기준에 사용될 수 있다.
- [0442] 60. CTU 크기가  $M$ 이고 기준 영역 크기가  $nM$ 이면, 기준 영역은 CTU 행/타일에서 가장 가까운 사용 가능한  $n-1$ 개의 CTU들이다.
- [0443] a. 하나의 예에서, 기준 영역 크기가  $128 \times 128$  또는  $256 \times 64$ 이고 CTU 크기가  $64 \times 64$ 이면, CTU 행에서 가장 가까운 사용 가능한 3개의 CTU들이 IBC 기준에 사용될 수 있다.
- [0444] b. 하나의 예에서, 기준 영역 크기가  $128 \times 128$  또는  $512 \times 32$ 이고 CTU 크기가  $32 \times 32$ 이면, CTU 행에서 가장 가까운 사용 가능한 15개의 CTU들이 IBC 기준에 사용될 수 있다.
- [0445] 61. CTU 크기가  $M$ 이고 VPDU 크기가  $kM$ 이며 기준 영역 크기가  $nM$ 이면, 기준 영역은 CTU 행/타일에서 가장 가까운 사용 가능한  $n-k$ 개의 CTU들이다.
- [0446] a. 하나의 예에서, CTU 크기가  $64 \times 64$ 이고 VPDU 크기 역시  $64 \times 64$ 이며 기준 영역 크기가  $128 \times 128$ 이면, CTU 행에서 가장 가까운 사용 가능한 3개의 CTU들이 IBC 기준에 사용될 수 있다.
- [0447] b. 하나의 예에서, CTU 크기가  $32 \times 32$ 이고 VPDU 크기가  $64 \times 64$ 이며 기준 영역 크기가  $128 \times 128$ 이면, CTU 행에서 가장 가까운 사용 가능한  $(16-4)=12$ 개의 CTU들이 IBC 기준에 사용될 수 있다.
- [0448] 62. IBC를 사용하여  $(x, y)$ 인 상-좌 모서리를 갖는  $w \times h$  블록에 대해, 메모리 재사용을 위해 어떤 영역으로부터의 기준 블록을 남겨두어야 하는 제약이 있는데, 여기서  $w$  및  $h$ 는 현재 블록의 폭 및 높이이다.
- [0449] a. 하나의 예에서, CTU 크기가  $128 \times 128$ 이고  $(x, y)=(m \times 64, n \times 64)$ 이면, 기준 블록은  $((m-2) \times 64, n \times 64)$ 에서 시작하는  $64 \times 64$  영역에 중첩될 수 없다.
- [0450] b. 하나의 예에서, CTU 크기가  $128 \times 128$ 이면, 기준 블록은  $(x-128, y)$ 인 상-좌 모서리의  $w \times h$  블록과 중첩될 수 없다.
- [0451] c. 하나의 예에서, CTU 크기가  $128 \times 128$ 이면,  $(x+Bv_x, y+Bv_y)$ 는  $(x-128, y)$ 인 상-좌 모서리의  $w \times h$  블록 내에 위치할 수 없는데, 여기서  $Bv_x$  및  $Bv_y$ 는 현재 블록의 블록 벡터를 표기한다.
- [0452] d. 하나의 예에서, CTU 크기가  $M \times M$ 이고 IBC 버퍼 크기가  $k \times M \times M$ 이면, 기준 블록은  $(x-k \times M, y)$ 인 상좌 블록과  $w \times h$  블록 내에서 중첩될 수 없는데, 여기서  $Bv_x$  및  $Bv_y$ 는 현재 블록의 블록 벡터를 표기한다.
- [0453] e. 하나의 예에서, CTU 크기가  $M \times M$ 이고 IBC 버퍼 크기가  $k \times M \times M$ 이면,  $(x+Bv_x, y+Bv_y)$ 는  $(x-k \times M, y)$ 인 상좌 블록과  $w \times h$  블록 내에 위치할 수 없는데, 여기서  $Bv_x$  및  $Bv_y$ 는 현재 블록의 블록 벡터를 표기한다.
- [0454] 63. CTU 크기가  $M \times M$ 이 아니고 기준 영역 크기가  $nM \times nM$ 이면, 기준 영역은 CTU 행에서 가장 가까운 사용 가능한  $n \times n-1$  CTU들이다.
- [0455] a. 하나의 예에서, 기준 영역 크기가  $128 \times 128$ 이고 CTU 크기가  $64 \times 64$ 이면, CTU 행에서 가장 가까운 사용 가능한 3개의 CTU들이 IBC 기준에 사용될 수 있다.
- [0456] b. 하나의 예에서, 기준 영역 크기가  $128 \times 128$ 이고 CTU 크기가  $32 \times 32$ 이면, CTU 행에서 가장 가까운 사용 가능한 15개의 CTU들이 IBC 기준에 사용될 수 있다.
- [0457] 64.  $(2m \times 64, 2n \times 64)$ 에서 시작되는  $64 \times 64$  블록 내, 즉  $128 \times 128$  CTU의 상-좌  $64 \times 64$  블록 내의 CU에 대해, 그 IBC 예측은  $((2m-2) \times 64, 2n \times 64)$ 에서 시작하는  $64 \times 64$  블록,  $((2m-1) \times 64, 2n \times 64)$ 에서 시작하는  $64 \times 64$  블록,  $((2m-1) \times 64, (2n+1) \times 64)$ 에서 시작하는  $64 \times 64$  블록, 및 현재  $64 \times 64$  블록 내의 재생 샘플들로부터 얻어질 수 있다.
- [0458] 65.  $((2m+1) \times 64, (2n+1) \times 64)$ 에서 시작되는  $64 \times 64$  블록 내, 즉  $128 \times 128$  CTU의 하-우  $64 \times 64$  블록 내의 CU에 대해, 그 IBC 예측은 현재  $128 \times 128$  CTU로부터 얻어질 수 있다.
- [0459] 66.  $((2m-1) \times 64, (2n+1) \times 64)$ 에서 시작되는  $64 \times 64$  블록 내, 즉  $128 \times 128$  CTU의 상-우  $64 \times 64$  블록 내의 CU에 대해, 그 IBC 예측은  $((2m-1) \times 64, 2n \times 64)$ 에서 시작하는  $64 \times 64$  블록,  $((2m-1) \times 64, (2n+1) \times 64)$ 에서 시작하는  $64 \times 64$  블록,  $(2m \times 64, 2n \times 64)$ 에서 시작하는  $64 \times 64$  블록, 및 현재  $64 \times 64$  블록 내의 재생 샘플들로부터 얻어질 수 있다.
- [0460] a. 이와는 달리,  $(2m \times 64, (2n+1) \times 64)$ 에서 시작하는  $64 \times 64$  블록이 재생되었다면, IBC 예측은

$((2m-1)*64, 2n*64)$ 에서 시작하는  $64x64$  블록,  $(2m*64, 2n*64)$ 에서 시작하는  $64x64$  블록,  $(2m*64, (2n+1)*64)$ 에서 시작하는  $64x64$  블록, 및 현재  $64x64$  블록 내의 재생 샘플들로부터 얻어질 수 있다.

[0461] 67.  $(2m*64, (2n+1)*64)$ 에서 시작되는  $64x64$  블록 내, 즉  $128x128$  CTU의 하-좌  $64x64$  블록 내의 CU에 대해, 그 IBC 예측은  $((2m-1)*64, (2n+1)*64)$ 에서 시작하는  $64x64$  블록,  $(2m*64, 2n*64)$ 에서 시작하는  $64x64$  블록,  $((2m+1)*64, 2n*64)$ 에서 시작하는  $64x64$  블록, 및 현재  $64x64$  블록 내의 재생 샘플들로부터 얻어질 수 있다.

[0462] a. 이와는 달리,  $((2m+1)*64, 2n*64)$ 에서 시작하는  $64x64$  블록이 재생되지 않았다면, IBC 예측은  $((2m-1)*64, 2n*64)$ 에서 시작하는  $64x64$  블록,  $((2m-1)*64, (2n+1)*64)$ 에서 시작하는  $64x64$  블록,  $(2m*64, 2n*64)$ 에서 시작하는  $64x64$  블록, 및 현재  $64x64$  블록 내의 재생 샘플들로부터 얻어질 수 있다.

[0463] 68. 현재 CU가 어느  $64x64$  블록들에 속하는가에 기반하여 기준 영역을 조정할 것이 제안된다.

[0464] a. 하나의 예에서,  $(y>>6)\&1 == 0$ 일 때  $(x,y)$ 에서 시작되는 CU에 대해,  $((x>>6<<6)-128, y>>6<<6)$  및  $((x>>6<<6)-64, y>>6<<6)$ 에서 시작하는 2개 또는 2개까지의 이전  $64x64$  블록들이 IBC 모드에 참조될 수 있다.

[0465] b. 하나의 예에서,  $(y>>6)\&1 == 1$ 일 때  $(x,y)$ 에서 시작되는 CU에 대해,  $((x>>6<<6)-64, y>>6<<6)$ 에서 시작되는 하나의 이전  $64x64$  블록이 IBC 모드에 참조될 수 있다.

[0466] 69.  $(x,y)$ 에서 시작되고 블록 벡터(BVx, BVy)를 가지는 블록에 대해,  $\text{isRec}(((x+Bv_x)>>6<<6)+128-((y+Bv_y)>>6)\&1)*64+(x\%64), ((y+Bv_y)>>6<<6)+(y\%64))$ 가 참이면 그 블록 벡터는 유효하지 않다.

[0467] a. 하나의 예에서, 블록이 루마 블록이다.

[0468] b. 하나의 예에서, 블록이 4:4:4 포맷의 크로마 블록이다.

[0469] c. 하나의 예에서, 블록이 루마 및 크로마 컴포넌트들 양자를 포함한다.

[0470] 70.  $(x,y)$ 에서 시작하고 블록 벡터(BVx, BVy)를 갖는 4:2:0 포맷의 크로마 블록에 대해,  $\text{isRec}(((x+Bv_x)>>5<<5)+64-((y+Bv_y)>>5)\&1)*32+(x\%32), ((y+Bv_y)>>5<<5)+(y\%32))$ 가 참이면, 이 블록 벡터는 유효하지 않다.

[0471] 71. 구성요소 c의 블록에 대한 BV가 유효하지 않은지 여부의 판단은 루마 샘플만을 점검하는 대신 구성요소 X의 샘플들의 사용 가능성에 의존할 수 있다.

[0472] a.  $(x,y)$ 에서 시작하고 블록 벡터(BVx, BVy)를 갖는 컴포넌트 c의 블록에 대해,  $\text{isRec}(c, ((x+Bv_x)>>6<<6)+128-((y+Bv_y)>>6)\&1)*64+(x\%64), ((y+Bv_y)>>6<<6)+(y\%64))$ 가 참이면, 그 블록 벡터는 유효하지 않은 것으로 취급될 수 있다.

[0473] i. 하나의 예에서, 블록이 루마 블록이다(예를 들어 c가 루마 컴포넌트 또는 RGB 코딩을 위한 G 컴포넌트).

[0474] ii. 하나의 예에서, 블록이 4:4:4 포맷의 크로마 블록이다(예를 들어 c가 cb 또는 cr 컴포넌트이거나, 또는 RGB 코딩을 위한 G 컴포넌트).

[0475] iii. 하나의 예에서, 예를 들어 루마 및 크로마 컴포넌트들 양자를 포함하는 블록 등 루마 및 크로마 컴포넌트들 양자에 대한 샘플들의 사용 가능성이 점검될 수 있다.

[0476] b. 구성요소 c의  $(x, y)$ 에서 시작하고 블록 벡터(BVx, BVy)를 갖는 4:2:0 포맷의 크로마 블록에 대해,  $\text{isRec}(c, ((x+Bv_x)>>5<<5)+64-((y+Bv_y)>>5)\&1)*32+(x\%32), ((y+Bv_y)>>5<<5)+(y\%32))$ 가 참이면, 이 블록 벡터는 유효하지 않은 것으로 취급될 수 있다.

[0477] c. 구성요소 c의  $(x, y)$ 에서 시작하고 블록 벡터(BVx, BVy)를 갖는 크로마 블록 또는 서브블록에 대해, 크로마 컴포넌트에 대한  $\text{isRec}(c, x+Bv_x+\text{Chroma\_CTU\_size}, y)$ 가 참이면 그 블록 벡터는 유효하지 않은 것으로 취급될 것인데, 여기서 Chroma\_CTU\_size는 크로마 컴포넌트에 대한 CTU 크기이다.

[0478] i. 하나의 예에서, 4:2:0 포맷에 대해, Chroma\_CTU\_size는 64일 수 있다.

[0479] ii. 하나의 예에서, 크로마 서브블록은 4:2:0 포맷의  $2x2$  블록이 될 수 있다.

[0480] iii. 하나의 예에서, 크로마 서브블록은 4:2:4 포맷의  $4x4$  블록이 될 수 있다.

- [0481] iv. 하나의 예에서, 크로마 서브블록은 루마 컴포넌트 내의 최소 CU 크기에 해당할 수 있다.
- [0482] 1. 이와는 달리, 크로마 서브블록이 크로마 구성요소에 대한 최소 CU 크기에 해당할 수 있다.
- [0483] 72. 이상에 언급된 모든 항목들에서, 기준 버퍼가 복수의  $M \times M$  블록들( $M=64$ )을 포함한다고 가정된다. 그러나 이는 기준 버퍼가 복수의  $N \times M$  블록들(예를 들어  $N=128$ ,  $M=64$ )을 포함하는 등 다른 경우들에 확장될 수 있다.
- [0484] 73. 이상에 언급된 모든 항목들에서, 기준 버퍼가 현재 블록과 동일한 브릭/타일/타일 그룹/슬라이스 내에 위치해야 한다는 추가적인 제약들이 적용될 수 있다.
- [0485] a. 하나의 예에서, 기준 버퍼의 일부가 현재 브릭/타일/타일 그룹/슬라이스 외부에 있으면, IBC의 사용이 디스에이블된다(disabled). IBC 관련 신호들 요소들의 시그널링이 스킵될(skipped) 수 있다.
- [0486] b. 이와는 달리, 기준 버퍼의 일부가 현재 브릭/타일/타일 그룹/슬라이스 외부에 있을 때, IBC가 여전히 하나의 블록에 대해 이네이بل될 수 있지만, 하나의 블록에 연계된 블록 벡터는 잔여(remaining) 기준 버퍼만을 가리킬 수 있다.
- [0487] 74. 가능하다면 CTU/CPB 행의 제1 VPDU 행에  $K_1$ 개의 가장 최근에 코딩된 VPDU를, 가능하다면 CTU/CPB 행의 제2 VPDU 행에  $K_2$ 개의 가장 최근에 코딩된 VPDU를 현재 VPDU를 제외하고 IBC에 대한 기준 영역으로 가질 것이 제안된다.
- [0488] a. 하나의 예에서,  $K_1$ 은 2와 같고  $K_2$ 는 1과 같다.
- [0489] b. 하나의 예에서, 이상의 방법은 CTU/CTB 크기가  $128 \times 128$ 이고 VPDU 크기가  $64 \times 64$ 일 때 적용될 수 있다.
- [0490] c. 하나의 예에서, 이상의 방법은 CTU/CTB 크기가  $64 \times 64$ 이고 VPDU 크기가  $64 \times 64$  및/또는  $32 \times 32$ 일 때 적용될 수 있다.
- [0491] d. 하나의 예에서, 이상의 방법은 CTU/CTB 크기가  $32 \times 32$ 이고 VPDU 크기가  $32 \times 32$  이하일 때 적용될 수 있다.
- [0492] 75. 이상의 방법들은 다른 스테이지(stage)들에 적용될 수 있다.
- [0493] a. 하나의 예에서, 블록 벡터(BV)들의 모듈러 연산(예를 들어  $a \bmod b$ )이 BV가 유효한지 여부를 결정하는 사용 가능성 점검 프로세스에서 호출(invoked)될 수 있다.
- [0494] b. 하나의 예에서, 블록 벡터(BV)들의 모듈러 연산(예를 들어  $a \bmod b$ )이 (예를 들어 루프-내 필터링 프로세스 전에) IBC 가상 버퍼 또는 재생 화면 버퍼 내의 (예를 들어 현재 샘플의 위치와 BV의 모듈러 연산) 결과에 따라 참조 샘플의 위치를 식별하도록 호출될 수 있다.
- [0495] **5. 실시예들(Embodiments)**
- [0496] **5.1 실시예 #1(Embodiment #1)**
- [0497] IBC를 위한 버퍼의 구현이 이하에 설명되는데:
- [0498] 버퍼 크기는  $128 \times 128$ 이다. CTU 크기 역시  $128 \times 128$ 이다. CTU 행의 첫 번째 CTU를 코딩하기 위해, (8비트 비디오 신호에 대해) 버퍼가 128로 초기화된다(initialized). CTU 행의  $k$ 번째 CTU를 코딩하기 위해, 버퍼는  $(k-1)$ 번째 CTU의 루프 필터링 이전의 재생(값)으로 초기화된다.
- [0499] 도 3은  $(x, y)$ 에서 시작되는 블록의 코딩의 예를 보인다.
- [0500] 코딩이 현재 CTU에 관련된  $(x, y)$ 에서 시작되면, 블록 벡터( $BV_x, BV_y$ ) =  $(x-x_0, y-y_0)$ 가 디코더로 전송되어 기준 블록이 IBC 버퍼 내의  $(x_0, y_0)$ 부터라고 표시한다. 이 블록의 폭과 높이가 각각  $w$ 와  $h$ 라고 가정한다. 이 블록의 코딩이 완료되면 IBC 버퍼 내의  $(x, y)$ 에서 시작하는  $w \times h$  영역이 루프 필터링 이전의 블록의 재생(값)으로 갱신될 것이다.
- [0501] **5.2 실시예 #2(Embodiment #2)**
- [0502] 도 4는 이전에 코딩된  $64 \times 4$  블록들을 선택하는 가능한 대체적 방법의 예를 보인다.

- [0503] **5.3 실시예 #3(Embodiment #3)**
- [0504] 도 5는 64x4 블록들의 코딩/디코딩 순서를 변경하는 가능한 대체적 방법의 예를 보인다.
- [0505] **5.4 실시예 #4(Embodiment #4)**
- [0506] 도 8은 64x64 블록들에 대한 디코딩 순서가 상에서 하로, 좌에서 우로일 때, 이전 64x4 블록들을 선택하는 다른 가능한 대체적 방법을 보인다.
- [0507] **5.5 실시예 #5(Embodiment #5)**
- [0508] 도 9는 이전에 코딩된 64x4 블록들을 선택하는 다른 가능한 대체적 방법을 보인다.
- [0509] **5.6 실시예 #6(Embodiment #6)**
- [0510] 도 11은 64x64 블록들에 대한 디코딩 순서가 상에서 하로, 좌에서 우로일 때, 이전 64x4 블록들을 선택하는 다른 가능한 대체적 방법을 보인다.
- [0511] **5.7 실시예 #7(Embodiment #7)**
- [0512] CTU 크기가  $W \times W$ 라고 가정하면, 디코더에서 크기  $m \times W$ 를 갖고 비트심도가 B인 IBC 버퍼의 구현은 다음과 같다.
- [0513] CTU 행의 디코딩 시작시, 버퍼를 값  $(1 \ll (B-1))$ 으로 초기화하고  $(x_b, y_b)$ 가  $(0, 0)$ 로 갱신되도록 시작점(starting point)을 설정한다.
- [0514] CU가 상-좌 모서리에 관련된  $(x, y)$ 에서 시작되고 디코딩될 크기  $w \times h$ 를 가지면,  $(x_b+x, y_b+y)$ 에서 시작되는  $w \times h$  크기의 영역이 비트심도가 B 비트에 정렬된 이후의 CU의 재생 화소값으로 갱신될 것이다.
- [0515] CTU가 디코딩된 다음,  $(x_b, y_b)$ 로 갱신된 시작점은  $((x_b+W) \bmod mW, 0)$ 로 설정될 것이다.
- [0516] CTU 상-좌 모서리에 관련된 어떤 화소  $(x, y)$ 에 대해 IBC CU를 블록 벡터(BVx, BVy)로 디코딩하면, 그 예측은 예측 신호들의 비트심도에 대한 비트심도 정렬 이후의 위치  $((x+BVx) \bmod mW, (y+BVy) \bmod W)$ 에서의 버퍼로부터 추출된다.
- [0517] 하나의 예에서, B가 7 또는 8로 설정되는 한편 그 블록의 출력/입력 비트심도는 10일 수 있다.
- [0518] **5.8 실시예 #8(Embodiment #8)**
- [0519] 화면의 상-좌 모서리와 블록 벡터(BVx, BVy)에 관련된  $(x, y)$ 에서 시작되는 루마 CU 또는 결합(joint) 루마/크로마 CU에 대해,  $\text{isRec}(((x+BVx) \gg 6 \ll 6) + 128 - (((y+BVy) \gg 6) \& 1) * 64 + (x \% 64), ((y+BVy) \gg 6 \ll 6) + (y \% 64))$ 가 참이면 그 블록 벡터는 유효하지 않다.
- [0520] 화면의 상-좌 모서리와 블록 벡터(BVx, BVy)에 관련된  $(x, y)$ 에서 시작되는 크로마 CU에 대해,  $\text{isRec}(((x+BVx) \gg 5 \ll 5) + 64 - (((y+BVy) \gg 5) \& 1) * 32 + (x \% 32), ((y+BVy) \gg 5 \ll 5) + (y \% 32))$ 가 참이면 그 블록 벡터는 유효하지 않다.
- [0521] **5.9 실시예 #9(Embodiment #9)**
- [0522] 화면의 상-좌 모서리와 블록 벡터(BVx, BVy)에 관련된  $(x, y)$ 에서 4:2:0 포맷으로 시작되는 크로마 블록 또는 서브블록에 대해,  $\text{isRec}(c, (x+BVx+64, y+BVy))$ 가 참이면 그 블록 벡터는 유효하지 않은데, 여기서 c는 크로마 컴포넌트이다.
- [0523] 화면의 상-좌 모서리와 블록 벡터(BVx, BVy)에 관련된  $(x, y)$ 에서 4:4:4 포맷으로 시작되는 크로마 블록 또는 서브블록에 대해,  $\text{isRec}(c, (x+BVx+128, y+BVy))$ 가 참이면 그 블록 벡터는 유효하지 않은데, 여기서 c는 크로마 컴포넌트이다.
- [0524] **5.10 실시예 #10(Embodiment #10)**
- [0525] 화면의 상-좌 모서리와 블록 벡터(BVx, BVy)에 관련된  $(x, y)$ 에서 시작되는 루마 CU 또는 결합 루마/크로마 CU에 대해,  $\text{isRec}(((x+BVx) \gg 6 \ll 6) + 128 - (((y+BVy) \gg 6) \& 1) * 64 + (x \% 64), ((y+BVy) \gg 6 \ll 6) + (y \% 64))$ 가 참이면 그 블록 벡터는 유효하지 않다.
- [0526] 화면의 상-좌 모서리와 블록 벡터(BVx, BVy)에 관련된  $(x, y)$ 에서 4:2:0 포맷으로 시작되는 크로마 블록 또는 서브블록에 대해,  $\text{isRec}(c, ((x+BVx) \gg 5 \ll 5) + 64 - (((y+BVy) \gg 5) \& 1) * 32 + (x \% 32), ((y+BVy) \gg 5 \ll 5) + (y \% 32))$ 가 참

이런 그 블록 벡터는 유효하지 않는데, 여기서 c는 크로마 컴포넌트이다.

[0527] **5.11 실시예 #11(Embodiment #11)**

[0528] 이 실시예는 현재의 VPDU를 제외하고 2개의 가장 최근 코딩된 VPDU들을 첫 번째 VPDU 행에 유지하고 하나의 가장 최근 코딩된 VPDU를 CTU/CTB 행의 두 번째 VPDU 행에 유지하는 구현을 강조하고 있다.

[0529] VPDU 코딩 순서가 상에서 하로 및 좌에서 우로일 때의 기준 영역이 도 13에 도시되어 있다.

[0530] VPDU 코딩 순서가 좌에서 우로 및 상에서 하로 이고 현재 VPDU가 화면 경계의 우측을 향하지 않을 때의 기준 영역이 도 14에 도시되어 있다.

[0531] VPDU 코딩 순서가 좌에서 우로 및 상에서 하로 이고 현재 VPDU가 화면 경계의 우측을 향할 때의 기준 영역이 도 15에 도시되어 있다.

[0532] 크기 w x h를 갖는 루마 블록(x, y)을 고려해보면, 블록 벡터(BVx, BVy)가 유효한지 여부는 다음 조건의 점검함으로써 구분될 수 있는데:

[0533]  $\text{isRec}(((x+Bv_x+128)>>6<<6) - (\text{ref}_y\&0x40) + (x\%64), ((y+Bv_y)>>6<<6) + (\text{ref}_y>>6 == y>>6)?(y\%64):0)$ , 여기서  $\text{ref}_y = (y\&0x40) ? (y+Bv_y) : (y+Bv_y-1)$ 이다.

[0534] 위 함수가 참을 회신하면, 블록 벡터(BVx, BVy)가 유효하지 않고, 그렇지 않으면 블록 벡터가 유효하다.

[0535] **5.12 실시예 #12(Embodiment #12)**

[0536] CTU 크기가 192x128이면, 크기 192x128을 갖는 가상 버퍼가 IBC에 대한 참조 샘플들을 추적하기 위해 유지된다.

[0537] 화면의 상-좌 모서리에 대한 샘플(x, y)은 버퍼의 상-좌 모서리에 대한 위치(x%192, y%128)에 연계된다. 다음 단계들은 IBC 참조를 위한 가상 버퍼에 연계된 샘플들의 사용 가능성을 마킹하는 방법을 보인다.

[0538] 화면의 상-좌 모서리에 대한 위치(xPrevVPDU, yPrevVPDU)는 가장 최근 디코딩된 VPDU의 상-좌 샘플을 나타내도록(stand for) 기록된다.

[0539] 1) VPDU 행의 디코딩 시작시, 버퍼의 모든 위치들은 사용 불가능한 것으로 마킹된다. (xPrevVPDU, yPrevVPDU)가 (0,0)으로 설정된다..

[0540] 2) VPDU의 첫 번째 CU의 디코딩 시작시,  $x = (x_{\text{PrevVPDU}} - 2W_{\text{VPDU}} + 2m_{\text{WVPDU}})\%(m_{\text{WVPDU}})$ , ...,  $((x_{\text{PrevVPDU}} - 2W_{\text{VPDU}} + 2m_{\text{WVPDU}})\%(m_{\text{WVPDU}}) - 1 + W_{\text{VPDU}})$ ; 이고  $y = y_{\text{PrevVPDU}}\%(n_{\text{HVPDU}})$ , ...,  $(y_{\text{PrevVPDU}}\%(n_{\text{HVPDU}}) - 1 + H_{\text{VPDU}})$ 인 위치(x, y)는 사용 불가능한 것으로 마킹된다. 이어서 (xPrevVPDU, yPrevVPDU)가 (xCU, yCU), 즉 화면에 대한 CU의 상-좌 위치로 설정된다.

[0541] 3) CU의 디코딩 후,  $x = x_{\text{CU}}\%(m_{\text{WVPDU}})$ , ...,  $(x_{\text{CU}} + x_{\text{CU\_width}} - 1)\%(m_{\text{WVPDU}})$ 이고  $y = y_{\text{CU}}\%(n_{\text{HVPDU}})$ , ...,  $(y_{\text{CU}} + y_{\text{CU\_height}} - 1)\%(n_{\text{HVPDU}})$ 인 위치(x, y)가 사용 가능한 것으로 마킹된다.

[0542] 4) 블록 벡터(xBV, yBV)를 갖는 IBC CU에 대해,  $x = (x_{\text{CU}} + x_{\text{BV}})\%(m_{\text{WVPDU}})$ , ...,  $(x_{\text{CU}} + x_{\text{BV}} + x_{\text{CU\_width}} - 1)\%(m_{\text{WVPDU}})$ 이고  $y = (y_{\text{CU}} + y_{\text{BV}})\%(n_{\text{HVPDU}})$ , ...,  $(y_{\text{CU}} + y_{\text{BV}} + y_{\text{CU\_height}} - 1)\%(n_{\text{HVPDU}})$ 인 어떤 위치(x, y)가 사용 불가능한 것으로 마킹되면, 그 블록 벡터는 유효하지 않은 것으로 간주된다.

[0543] 도 16은 버퍼 상태를 화면 내의 VPDU 디코딩 상태와 함께 보인다.

[0544] **5.13 실시예 #13(Embodiment #13)**

[0545] CTU 크기가 128x128 또는 CTU 크기가 VPDU 크기보다 크면(예를 들어 현재 설계에서 64x64) 또는 CTU 크기가 VPDU 크기보다 크면(예를 들어 현재 설계에서 64x64), IBC에 대한 참조 샘플들을 추적하도록 크기 128x128을 갖는 가상 버퍼가 유지된다. 이하에서,  $a < 0$ ,  $(a \% b)$ 가  $(a/b)*b$ 로 정의될 때  $\text{floor}(c)$ 는 c보다 크지 않은 최대 정수를 회신한다.

[0546] 화면의 상-좌 모서리에 대한 샘플(x, y)은 버퍼의 상-좌 모서리에 대한 위치(x%192, y%128)에 연계된다. 이하의 단계들은 IBC 참조를 위한 가상 버퍼에 연계된 샘플들의 사용 가능성을 마킹하는 방법을 보인다.

[0547] 화면의 상-좌 모서리에 대한 위치(xPrevVPDU, yPrevVPDU)가 가장 최근에 디코딩된 VPDU의 상-좌 샘플을 나타내도록 기록된다.

[0548] 1) VPDU 행의 디코딩 시작시, 버퍼의 모든 위치들이 사용 불가능한 것으로 마킹된다.

(xPrevVPDU, yPrevVPDU)가 (0,0)으로 설정된다.

[0549] 2) VPDU의 첫 번째 CU의 디코딩 시작시,

[0550] a.  $yPrevVPDU \% 64$ 가 0이면,  $x = (xPrevVPDU - 128) \% 192, \dots, ((xPrevVPDU - 128) \% 192) + 63$ ;이고  $y = yPrevVPDU \% 128, \dots, (yPrevVPDU \% 128) + 63$ 인 위치(x, y)들이 사용 불가능한 것으로 마킹된다. 이어서 (xPrevVPDU, yPrevVPDU)가 (xCU, yCU), 즉 화면에 대한 CU의 좌-상 위치로 설정된다.

[0551] b. 그렇지 않으면,  $x = (xPrevVPDU - 64) \% 192, \dots, ((xPrevVPDU - 64) \% 192) + 63$ ;이고  $y = yPrevVPDU \% 128, \dots, (yPrevVPDU \% 128) + 63$ 인 위치 (x, y)가 사용 불가능한 것으로 마킹된다. 이어서, (xPrevVPDU, yPrevVPDU)가 (xCU, yCU), 즉 화면에 대한 CU의 좌-상 위치로 설정된다.

[0552] 3) CU의 디코딩 후,  $x = xCU \% 192, \dots, (xCU + CU\_width - 1) \% 192$ 이고  $y = yCU \% 128, \dots, (yCU + CU\_height - 1) \% 128$ 인 위치(x, y)가 사용 가능한 것으로 마킹된다.

[0553] 4) 블록 벡터(xBV, yBV)를 갖는 IBC CU에 대해,  $x = (xCU + xBV) \% 192, \dots, (xCU + xBV + CU\_width - 1) \% 192$ 이고  $y = (yCU + yBV) \% 128, \dots, (yCU + yBV + CU\_height - 1) \% 128$ 인 어떤 위치가 사용 불가능한 것으로 마킹되면, 그 블록 벡터는 유효하지 않은 것으로 간주된다.

[0554] S가 128이 아닐 때 CTU 크기가 SxS라면, Wbuf가 128\*128/S와 동일한 것으로 한다. IBC를 위한 참조 샘플들을 추적하기 위해 크기 WbufxS를 갖는 가상 버퍼가 유지된다. 이러한 경우 VPDU 크기가 CTU 크기와 같다.

[0555] 화면의 상-좌 모서리에 대한 위치(xPrevVPDU, yPrevVPDU)가 가장 최근에 디코딩된 VPDU의 좌-상 샘플을 나타내도록 기록된다.

[0556] 1) VPDU 행의 디코딩 시작시, 버퍼의 모든 위치들이 사용 불가능한 것으로 마킹된다. (xPrevVPDU, yPrevVPDU)가 (0,0)으로 설정된다.

[0557] 2) VPDU의 첫 번째 CU의 디코딩 시작시,  $x = (xPrevVPDU - W_{buf} * S) \% S, \dots, ((xPrevVPDU - W_{buf} * S) \% S) + S - 1$ ;이고  $y = yPrevVPDU \% S, \dots, (yPrevVPDU \% S) + S - 1$ 인 위치(x, y)들이 사용 불가능한 것으로 마킹된다. 이어서 (xPrevVPDU, yPrevVPDU)가 (xCU, yCU), 즉 화면에 대한 CU의 좌-상 위치로 설정된다.

[0558] 3) CU의 디코딩 후,  $x = xCU \% (W_{buf}), \dots, (xCU + CU\_width - 1) \% (W_{buf})$ 이고  $y = yCU \% S, \dots, (yCU + CU\_height - 1) \% S$ 인 위치(x, y)가 사용 가능한 것으로 마킹된다.

[0559] 4) 블록 벡터(xBV, yBV)를 갖는 IBC CU에 대해,  $x = (xCU + xBV) \% (W_{buf}), \dots, (xCU + xBV + CU\_width - 1) \% (W_{buf})$ 이고  $y = (yCU + yBV) \% S, \dots, (yCU + yBV + CU\_height - 1) \% S$ 인 위치(x, y)가 사용 불가능한 것으로 마킹되면 그 블록 벡터는 유효하지 않은 것으로 간주된다.

#### [0560] 5.14 실시예 #14(Embodiment #14)

[0561] CTU 크기가 128x128이거나 CTU 크기가 VPDU 크기보다 크면(예를 들어 현재 설계에서 64x64), IBC에 대한 참조 샘플들을 추적하도록 크기 256x128을 갖는 가상 버퍼가 유지된다. 이하에서,  $a < 0$ ,  $(a \% b)$ 가  $(a/b) * b$ 로 정의될 때  $\text{floor}(c)$ 는 c보다 크지 않은 최대 정수를 회신한다.

[0562] 화면의 상-좌 모서리에 대한 샘플(x, y)이 버퍼의 상-좌 모서리에 대한 위치( $x \% 256, y \% 128$ )에 연계된다. 다음 단계들은 IBC 기준을 위한 가상 버퍼에 연계된 샘플들의 사용 가능성을 마킹하는 방법을 보인다.

[0563] 화면의 상-좌 모서리에 대한 위치(xPrevVPDU, yPrevVPDU)가 가장 최근에 디코딩된 VPDU를 나타내도록 기록된다.

[0564] 1) VPDU 행의 디코딩 시작시, 버퍼의 모든 위치들이 사용 불가능한 것으로 마킹된다. (xPrevVPDU, yPrevVPDU)가 (0,0)으로 설정된다.

[0565] 2) VPDU의 첫 번째 CU의 디코딩 시작시,

[0566] a.  $yPrevVPDU \% 64$ 가 0이면,  $x = (xPrevVPDU - 128) \% 256, \dots, ((xPrevVPDU - 128) \% 256) + 63$ ;이고  $y = yPrevVPDU \% 128, \dots, (yPrevVPDU \% 128) + 63$ 인 위치(x, y)들이 사용 불가능한 것으로 마킹된다. 이어서 (xPrevVPDU, yPrevVPDU)가 (xCU, yCU), 즉 화면에 대한 CU의 상-좌 위치로 설정된다.

[0567] b. 그렇지 않으면,  $x = (xPrevVPDU - 64) \% 256, \dots, ((xPrevVPDU - 64) \% 256) + 63$ ;이고  $y = yPrevVPDU \% 128, \dots, (yPrevVPDU \% 128) + 63$ 인 위치(x, y)들이 사용 불가능한 것으로 마킹된다. 이어서

(xPrevVPDU, yPrevVPDU)가 (xCU, yCU), 즉 화면에 대한 CU의 상-좌 위치로 설정된다.

[0568] 3) CU의 디코딩 후,  $x = xCU\%256, \dots, (xCU+CU\_width-1)\%256$ 이고  $y = yCU\%128, \dots, (yCU+CU\_height-1)\%128$ 인 위치(x, y)가 사용 가능한 것으로 설정된다.

[0569] 4) 블록 벡터(xBV, yBV)를 갖는 IBC CU에 대해  $x = (xCU+xBV)\%256, \dots, (xCU+xBV+CU\_width-1)\%256$ 이고  $y = (yCU+yBV)\%128, \dots, (yCU+yBV+CU\_height-1)\%128$ 인 위치(x, y)가 사용 불가능한 것으로 마킹되면 그 블록 벡터는 유효하지 않은 것으로 간주된다.

[0570] CTU 크기가 128x128이 아니거나 64x64보다 작거나 64x64보다 작거나, 이전 실시예, 즉 실시예 #14에서와 동일한 프로세스가 적용된다.

#### [0571] 5.15 실시예 #15(Embodiment #15)

[0572] IBC 기준 사용 가능성 마킹 프로세스(IBC reference availability marking process)가 다음과 같이 설명된다. 변경은 이 명세서에서 굵게 밑줄 친 이탤릭체 텍스트로 표시된다.

#### [0573] 7.3.7.1 일반적 슬라이스 데이터 선택스(General slice data syntax)

표 4

| slice_data() {                                                         | descriptor |
|------------------------------------------------------------------------|------------|
| for(i = 0; i < NumBricksInCurrSlice; i++) {                            |            |
| CtbAddrInBs = FirstCtbAddrBs[SliceBrickIdx[i]]                         |            |
| for(j = 0; j < NumCtusInBrick[SliceBrickIdx[i]]; j++, CtbAddrInBs++) { |            |
| if((j % BrickWidth[SliceBrickIdx[i]]) == 0) {                          |            |
| NumHmvpCand = 0                                                        |            |
| NumHmvpIbcCand = 0                                                     |            |
| xPrevVPDU = 0                                                          |            |
| yPrevVPDU = 0                                                          |            |
| if(CtbSizeY == 128)                                                    |            |
| reset_ibc_isDecoded(0, 0, 256, CtbSizeY,                               |            |
| BufWidth, BufHeight)                                                   |            |
| else                                                                   |            |
| reset_ibc_isDecoded(0, 0, 128*128/CtbSizeY,                            |            |
| CtbSizeY, BufWidth, BufHeight)                                         |            |
| }                                                                      |            |
| CtbAddrInRs = CtbAddrBsToRs[CtbAddrInBs]                               |            |
| ... ..                                                                 |            |
| reset_ibc_isDecoded(x0, y0, w, h, BufWidth, BufHeight) {               | descriptor |
| if(x0 >= 0)                                                            |            |
| for (x = x0 % BufWidth; x < x0 + w; x+=4)                              |            |
| for (y = y0 % BufHeight; y < y0 + h; y+=4)                             |            |
| isDecoded[x >> 2][y >> 2] = 0                                          |            |
| }                                                                      |            |

[0575] BufWidth가 (CtbSizeY==128)?256:(128\*128/CtbSizeY)와 같고 BufHeight가 CtbSizeY와 같다.

#### [0576] 7.3.7.5 코딩 유닛 선택스(Coding unit syntax)

표 5

| coding_unit(x0, y0, cbWidth, cbHeight, treeType) {                                          | descriptor |
|---------------------------------------------------------------------------------------------|------------|
| if(treeType != DUAL_TREE_CHROMA && (CtbSizeY == 128) && (x0 % 64) == 0 && (y0 % 64) == 0) { |            |
| for(x = x0; x < x0 + cbWidth; x += 64)                                                      |            |
| for(y = y0; y < y0 + cbHeight; y += 64)                                                     |            |
| if((yPrevVPDU % 64) == 0)                                                                   |            |

|                                                                                                                                                                                                                                                                                                               |       |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| <u>reset_ibc_isDecoded(xPrevVPDU - 128, yPrevVPDU, 64, 64, BufWidth, BufHeight)</u>                                                                                                                                                                                                                           |       |
| <u>else</u>                                                                                                                                                                                                                                                                                                   |       |
| <u>reset_ibc_isDecoded(xPrevVPDU - 64, yPrevVPDU, 64, 64, BufWidth, BufHeight)</u>                                                                                                                                                                                                                            |       |
| <u>xPrevVPDU = x0</u>                                                                                                                                                                                                                                                                                         |       |
| <u>yPrevVPDU = y0</u>                                                                                                                                                                                                                                                                                         |       |
| <u>}</u>                                                                                                                                                                                                                                                                                                      |       |
| <u>if(treeType != DUAL_TREE_CHROMA &amp;&amp; (CtbSizeY &lt; 128) &amp;&amp; (x0 % CtbSizeY) == 0 &amp;&amp; (y0 % CtbSizeY) == 0) {</u>                                                                                                                                                                      |       |
| <u>reset_ibc_isDecoded(xPrevVPDU - (128*128/CtbSizeY - CtbSizeY), yPrevVPDU, 64, 64, BufWidth, BufHeight)</u>                                                                                                                                                                                                 |       |
| <u>xPrevVPDU = x0</u>                                                                                                                                                                                                                                                                                         |       |
| <u>yPrevVPDU = y0</u>                                                                                                                                                                                                                                                                                         |       |
| <u>}</u>                                                                                                                                                                                                                                                                                                      |       |
| <u>if(slice_type != I    sps_ibc_enabled_flag) {</u>                                                                                                                                                                                                                                                          |       |
| <u>if((treeType != DUAL_TREE_CHROMA &amp;&amp; !(cbWidth == 4 &amp;&amp; cbHeight == 4 &amp;&amp; !sps_ibc_enabled_flag))</u>                                                                                                                                                                                 |       |
| <u>cu_skip_flag[x0][y0]</u>                                                                                                                                                                                                                                                                                   | ae(v) |
| <u>if(cu_skip_flag[x0][y0] == 0 &amp;&amp; slice_type != I &amp;&amp; !(cbWidth == 4 &amp;&amp; cbHeight == 4))</u>                                                                                                                                                                                           |       |
| <u>pred_mode_flag</u>                                                                                                                                                                                                                                                                                         | ae(v) |
| <u>if(((slice_type == I &amp;&amp; cu_skip_flag[x0][y0] == 0)    (slice_type != I &amp;&amp; (CuPredMode[x0][y0] != MODE_INTRA    (cbWidth == 4 &amp;&amp; cbHeight == 4 &amp;&amp; cu_skip_flag[x0][y0] == 0)))) &amp;&amp; sps_ibc_enabled_flag &amp;&amp; (cbWidth != 128 &amp;&amp; cbHeight != 128))</u> |       |
| <u>pred_mode_ibc_flag</u>                                                                                                                                                                                                                                                                                     | ae(v) |
| <u>}</u>                                                                                                                                                                                                                                                                                                      |       |
| <u>...</u>                                                                                                                                                                                                                                                                                                    |       |

8.6.2 IBC 블록들에 대한 모션 벡터 컴포넌트들의 도출 프로세스(Derivation process for motion vector components for IBC blocks)8.6.2.1 개요(General)

...

항목 8.6.3.2의 블록 벡터 유효성 점검 프로세스가 블록 벡터(mvL)를 입력으로(with) 호출될 때, isBVvalid가 참이어야 한다는 것이 비트스트림 적합성의 요건이다.

...

8.6.3 IBC 블록들의 디코딩 프로세스(Decoding process for ibc blocks)

8.6.3.1 개요(General)

이 프로세스는 IBC 예측 모드로 코딩된 코딩 유닛을 디코딩할 때 요구된다.

이 프로세스에 대한 입력들은:

- 현재 화면에 상-좌 루마 샘플에 대한 현재 코딩 블록의 좌-상 샘플을 규정하는 루마 위치(xCb, yCb),
- 루마 샘플들 내의 현재 코딩 블록의 높이를 규정하는 변수 cbHeight.
- 수평 및 수직 방향의 루마 코딩 서브블록들의 수를 규정하는 변수 numSbX 및 numSbY,
- 모션 벡터들 mv[xSbIdx][ySbIdx] with xSbIdx = 0..numSbX - 1, 및 ySbIdx = 0..numSbY - 1,
- 현재 블록의 칼라 컴포넌트 인덱스를 규정하는 변수 cIdx.
- $(nIbcBufW) \times (ctbSize)$  배열 ibcBuf

- [0592]  $xSbIdx = 0..numSbX - 1$ 이고  $ySbIdx = 0..numSbY - 1$ 인 서브블록 인덱스( $xSbIdx$ ,  $ySbIdx$ )에서의 각 코딩 서브블록에 대해 다음이 적용된다:
- [0593] - 현재 화면의 상-좌 루마 샘플에 대한 현재 코딩 서브블록의 상-좌 샘플을 규정하는 루마 위치( $xSb$ ,  $ySb$ )는 다음과 같이 도출된다:
- [0594]  $(xSb, ySb) = (xCb + xSbIdx * sbWidth, yCb + ySbIdx * sbHeight)$  (8-913)
- [0595]  $cIdx$ 가 0이면  $nIbcBufW$ 가  $ibcBufferWidth$ 로 설정되고, 그렇지 않으면  $nIbcBufW$ 가  $(ibcBufferWidth / SubWidthC)$ 로 설정된다. 다음이 적용된다:
- [0596]  $predSamples[xSb + x][ySb + y] = ibcBuf[(xCb + x + (mv[xSb][ySb][0] >> 4)) \% nIbcRefW][ySb + y + (mv[xSb][ySb][1] >> 4)]$
- [0597] 여기서  $x = 0..sbWidth - 1$ 이고  $y = 0..sbHeight - 1$ .
- [0598] ...
- [0599] 8.6.3.2 블록 벡터 유효성 점검 프로세스(Block vector validity checking process)
- [0600] 이 프로세스에 대한 입력은:
- [0601] - 현재 화면의 상-좌 루마 샘플에 대한 현재 코딩 블록의 상-좌 샘플을 규정하는 루마 위치( $xCb$ ,  $yCb$ ).
- [0602] - 루마 샘플들 내의 현재 코딩 블록의 폭을 규정하는 변수  $cbWidth$ .
- [0603] - 루마 샘플들 내의 현재 코딩 블록의 높이를 규정하는 변수  $cbHeight$ .
- [0604] - 수평 및 수직 방향의 루마 코딩 서브블록들의 수를 규정하는 변수  $numSbX$  및  $numSbY$ .
- [0605] -  $xSbIdx = 0..numSbX - 1$ 이고  $ySbIdx = 0..numSbY - 1$ 인 블록 벡터  $mv[xSbIdx][ySbIdx]$ .
- [0606] - 현재 블록의 칼라 컴포넌트를 규정하는 변수  $cid$ .
- [0607] -  $(nIbcBufW) \times (ctbSize)$  배열  $ibcBuf$
- [0608] 이 프로세스의 출력들은 블록 벡터가 유효한지 여부를 표시하는 플래그  $isBVvalid$ 이다.
- [0609] 다음이 적용된다
- [0610] 1.  $isBVvalid$ 이 참(true)으로 설정된다.
- [0611] 2.  $((yCb \& (ctbSize - 1)) + mv[0][0][1] + cbHeight) > ctbSize$ 이면,  $isBVvalid$ 가 거짓(false)으로 설정된다.
- [0612] 3. 그렇지 않으면,  $xSbIdx = 0..numSbX - 1$ 이고  $ySbIdx = 0..numSbY - 1$ 인 각 서브블록 인덱스( $xSbIdx$ ,  $ySbIdx$ )에 대해,  $ibcBuf$ 의 상-좌 루마 샘플들에 대한 그 위치가 도출되는데:
- [0613]  $xTL = (xCb + xSbIdx * sbWidth + mv[xSbIdx][ySbIdx][0]) \& (nIbcBufW - 1)$
- [0614]  $yTL = (yCb \& (ctbSize - 1)) + ySbIdx * sbHeight + mv[xSbIdx][ySbIdx][1]$
- [0615]  $xBR = (xCb + xSbIdx * sbWidth + sbWidth - 1 + mv[xSbIdx][ySbIdx][0]) \& (nIbcBufW - 1)$
- [0616]  $yBR = ((yCb \& (ctbSize - 1)) + ySbIdx * sbHeight + sbHeight - 1 + mv[xSbIdx][ySbIdx][1])$
- [0617]  $(isDecoded[xTL >> 2][yTL >> 2] == 0)$  또는  $(isDecoded[xBR >> 2][yTL >> 2] == 0)$  또는  $(isDecoded[xBR >> 2][yBR >> 2] == 0)$ 이면,  $isBVvalid$ 가 거짓으로 설정된다.
- [0618] 8.7.5 화면 재생 프로세스(Picture reconstruction process)
- [0619] 8.7.5.1 개요(General)
- [0620] 이 프로세스에 대한 입력들은:

- [0621] - 현재 화면 구성요소의 상-좌 샘플에 대한 현재 블록의 상-좌 샘플을 규정하는 위치(xCurr, yCurr),
- [0622] - 각각 현재 블록의 폭과 높이를 규정하는 변수 nCurrSw 및 nCurrSh,
- [0623] - 현재 블록의 칼라 컴포넌트를 규정하는 변수 cIdx,
- [0624] - 현재 블록의 예측 샘플들을 규정하는 (nCurrSw)x(nCurrSh) 배열 predSamples,
- [0625] - 현재 블록의 잔차 샘플(residual sample)들을 규정하는 (nCurrSw)x(nCurrSh) 배열 resSamples.
- [0626] 이 프로세스의 출력은
- [0627] - 재생 화면 샘플 배열 recSamples.
- [0628] - IBC 기준 배열 ibcBuf.
- [0629] ...
- [0630] ibcBuf의 폭을 nIbcBufW로 표기하면, 다음이 적용된다:
- [0631] 
$$ibcBuf[(xCurr + i) \& (nIbcBufW - 1)][(yCurr + j) \& (ctbSize - 1)] = recSamples[xCurr + i][yCurr + j]$$
- [0632] 여기서  $i = 0..nCurrSw - 1, j = 0..nCurrSh - 1$ .
- [0633] **5.16 실시예 #16(Embodiment #16)**
- [0634] 이는 다음 변경들을 제외하면 이전 실시예와 동일하다.

**표 6**

|        |                                                                        |                   |
|--------|------------------------------------------------------------------------|-------------------|
| [0635] | slice_data() {                                                         | descriptor        |
|        | for(i = 0; i < NumBricksInCurrSlice; i++) {                            |                   |
|        | CtbAddrInBs = FirstCtbAddrBs[SliceBrickIdx[i]]                         |                   |
|        | for(j = 0; j < NumCtusInBrick[SliceBrickIdx[i]]; j++, CtbAddrInBs++) { |                   |
|        | if((j % BrickWidth[SliceBrickIdx[i]]) == 0) {                          |                   |
|        | NumHmvpCand = 0                                                        |                   |
|        | NumHmvpIbcCand = 0                                                     |                   |
|        | <u>xPrevVPDU = 0</u>                                                   |                   |
|        | <u>yPrevVPDU = 0</u>                                                   |                   |
|        | <u>if(CtbSizeY == 128)</u>                                             |                   |
|        | <u>reset_ibc_isDecoded(0, 0, 192, CtbSizeY,</u>                        |                   |
|        | <u>BufWidth, BufHeight)</u>                                            |                   |
|        | <u>else</u>                                                            |                   |
|        | <u>reset_ibc_isDecoded(0, 0,</u>                                       |                   |
|        | <u>128*128/CtbSizeY, CtbSizeY, BufWidth, BufHeight)</u>                |                   |
|        | }                                                                      |                   |
|        | CtbAddrInRs = CtbAddrBsToRs[CtbAddrInBs]                               |                   |
|        | .....                                                                  |                   |
|        | <u>reset_ibc_isDecoded(x0, y0, w, h, BufWidth, BufHeight) {</u>        | <u>Descriptor</u> |
|        | <u>if(x0 &gt;= 0)</u>                                                  |                   |
|        | <u>for (x = x0 % BufWidth; x &lt; x0 + w; x+=4)</u>                    |                   |
|        | <u>for (y = y0 % BufHeight; y &lt; y0 + h; y+=4)</u>                   |                   |
|        | <u>isDecoded[x &gt;&gt; 2][y &gt;&gt; 2] = 0</u>                       |                   |
|        | <u>}</u>                                                               |                   |

[0636] BufWidth는 (CtbSizeY==128)?192:(128\*128/CtbSizeY)와 같고 BufHeight는 CtbSizeY와 같다.

[0637] **5.17 실시예 #17(Embodiment #17)**

[0638] 이 명세서에서 일부 샘플들에서의 변경은 굵고 밑줄 친 이탤릭체 텍스트로 표시된다.

### [0639] 7.3.7 슬라이스 데이터 선택스(Slice data syntax)

#### [0640] 7.3.7.1 일반적 슬라이스 데이터 선택스(General slice data syntax)

#### 표 7

| [0641] | slice_data() {                                                                             | Descriptor |
|--------|--------------------------------------------------------------------------------------------|------------|
|        | for(i = 0; i < NumBricksInCurrSlice; i++) {                                                |            |
|        | CtbAddrInBs = FirstCtbAddrBs[SliceBrickIdx[i]]                                             |            |
|        | for(j = 0; j < NumCtusInBrick[SliceBrickIdx[i]]; j++, CtbAddrInBs++) {                     |            |
|        | if((j % BrickWidth[SliceBrickIdx[i]]) == 0) {                                              |            |
|        | NumHmvpCand = 0                                                                            |            |
|        | NumHmvpIbcCand = 0                                                                         |            |
|        | resetIbcBuf = 1                                                                            |            |
|        | }                                                                                          |            |
|        | CtbAddrInRs = CtbAddrBsToRs[CtbAddrInBs]                                                   |            |
|        | coding_tree_unit()                                                                         |            |
|        | if(entropy_coding_sync_enabled_flag &&<br>((j + 1) % BrickWidth[SliceBrickIdx[i]] == 0)) { |            |
|        | end_of_subset_one_bit /* equal to 1 */                                                     | ae(v)      |
|        | if(j < NumCtusInBrick[SliceBrickIdx[i]] - 1)                                               |            |
|        | byte_alignment()                                                                           |            |
|        | }                                                                                          |            |
|        | }                                                                                          |            |
|        | if(!entropy_coding_sync_enabled_flag) {                                                    |            |
|        | end_of_brick_one_bit /* equal to 1 */                                                      | ae(v)      |
|        | if(i < NumBricksInCurrSlice - 1)                                                           |            |
|        | byte_alignment()                                                                           |            |
|        | }                                                                                          |            |
|        | }                                                                                          |            |

#### [0642] 7.4.8.5 코딩 유닛 시맨틱들(Coding unit semantics)

[0643] 다음 모든 조건들이 참이면, 공유된 병합 후보 목록 영역(shared merging candidate list region)에 대한 사용 이력 기반(history-based) 모션 벡터 예측변수 목록이 NumHmvpSmrIbcCand를 NumHmvpIbcCand와 같도록 설정하고,  $i = 0..NumHmvpIbcCand - 1$ 에 대해 HmvpSmrIbcCandList[i]가 HmvpIbcCandList[i]와 같도록 설정함으로써 갱신되는데:

[0644] - IsInSmr[x0][y0]이 TRUE이다.

[0645] - SmrX[x0][y0]이 x0이다.

[0646] - SmrY[x0][y0]이 y0이다.

[0647] 다음 할당(assignment)들이  $x = x0..x0 + cbWidth - 1$  및  $y = y0..y0 + cbHeight - 1$ 에 대해 이뤄진다:

[0648]  $CbPosX[x][y] = x0$  (7-135)

[0649]  $CbPosY[x][y] = y0$  (7-136)

[0650]  $CbWidth[x][y] = cbWidth$  (7-137)

[0651]  $CbHeight[x][y] = cbHeight$  (7-138)

[0652]  $vSize$ 를  $\min(ctbSize, 64)$ 로,  $wIbcBuf$ 를  $(128*128/ctbSize)$ 로 설정한다. 이에 따라  $ibcBuf$ 의 폭 및 높이는  $wIbcBuf$  및  $ctbSize$ 이다.

[0653]  $refreshIbcBuf$ 가 1이면 다음이 적용된다

[0654]  $x = x0..x0 + wIbcBuf - 1$  및  $y = y0..y0 + ctbSize - 1$ 에 대해  $ibcBuf[x \% wIbcBuf][y \% ctbSize] = -1$

[0655]  $resetIbcBuf = 0$

[0656]  $(x0 \% vSize)$  가 0이고  $(y0 \% vSize)$  가 0이면, for  $x = x0..x0 + vSize - 1$  및  $y = y0..y0 + vSize - 1$ 에 대해 다음이 적용된다

[0657]  $ibcBuf[x \% wIbcBuf][y \% ctbSize] = -1$

[0658] 8.6.2 IBC 블록들에 대한 모션 벡터의 도출 프로세스(Derivation process for motion vector components for IBC blocks)

[0659] 8.6.2.1 개요(General)

[0660] 이 프로세스에 대한 입력들은:

[0661] - 현재 화면의 상-좌 루마 샘플에 대한 현재 루마 코딩 블록의 상-좌 샘플의 루마 위치(xCb, yCb),

[0662] - 루마 샘플들 내의 현재 코딩 블록의 폭을 규정하는 변수 cbWidth,

[0663] - 루마 샘플들 내의 현재 코딩 블록의 높이를 규정하는 변수 cbHeight

[0664] 이 프로세스의 출력들은:

[0665] - 1/16 프랙셔널 샘플 정확도(fractional-sample accuracy)의 루마 모션 벡터(mvL).

[0666] 루마 모션 벡터(mvL)는 다음과 같이 도출되는데:

[0667] - 8.6.2.2항에 규정된 IBC 모션 벡터 예측에 대한 도출 프로세스는 루마 위치(xCb, yCb), 변수 cbWidth 및 cbHeight 입력들로 호출되는데, 출력은 루마 모션 벡터(mvL)이다.

[0668] - general\_merge\_flag[xCb][yCb]가 0이면 다음이 적용된다:

[0669] 1. 변수 mvd는 다음과 같이 도출된다:

[0670]  $mvd[0] = MvdL0[xCb][yCb][0]$  (8-883)

[0671]  $mvd[1] = MvdL0[xCb][yCb][1]$  (8-884)

[0672] 2. 8.5.2.14항에 규정된 모션 벡터들에 대한 반올림 프로세스(rounding process)가 mvL로 설정된 mvX와, MvShift + 2로 설정된 rightShift와, 및 MvShift + 2로 설정된 rightShift를 입력으로 호출되고, 반올림된 mvL을 출력으로 한다.

[0673] 3. 루마 모션 벡터(mvL)가 다음과 같이 수정되는데:

[0674]  $u[0] = (mvL[0] + mvd[0] + 2^{18}) \% 2^{18}$  (8-885)

[0675]  $mvL[0] = (u[0] \geq 2^{17})?(u[0] - 2^{18}) : u[0]$  (8-886)

[0676]  $u[1] = (mvL[1] + mvd[1] + 2^{18}) \% 2^{18}$  (8-887)

[0677]  $mvL[1] = (u[1] \geq 2^{17})?(u[1] - 2^{18}) : u[1]$  (8-888)

[0678] 주목 1 - 위에 규정된 mvL[0] 및 mvL[1]의 결과 값들은 항상 양단을 포함하여  $-2^{17}$  내지  $2^{17} - 1$ 의 범위 내에 있을 것이다.

[0679] 8.6.2.6항에 규정된 사용이력 기반 모션 벡터 예측변수 목록의 갱신 프로세스가 루마 모션 벡터(mvL)를 입력으로(with) 호출된다.

[0680] 루마 블록 벡터(mvL)가 다음 제약들을 준수하는 것이 비트스트림 적합성의 요건인데:

- [0678]  $((yCb + (mvL[1] >> 4)) \% wIbcBuf) + cbHeight$ 가  $ctbSize$  이하여야 한다
- [0679]  $x = xCb..xCb + cbWidth - 1$  및  $y = yCb..yCb + cbHeight - 1$ 에 대해,  $ibcBuf[(x + (mvL[0] >> 4)) \% wIbcBuf][(y + (mvL[1] >> 4)) \% ctbSize]$ 가 -1이 아니어야 한다.
- [0680] **8.7.5 화면 재생 프로세스(Picture reconstruction process)**
- [0681] **8.7.5.1 개요(General)**
- [0682] 이 프로세스에 대한 입력들은:
- [0683] - 현재 화면 컴포넌트의 상-좌 샘플에 대한 현재 블록의 상-좌 샘플을 규정하는 위치(xCurr, yCurr),
- [0684] - 각각 현재 블록의 폭 및 높이를 규정하는 변수 nCurrSw 및 nCurrSh,
- [0685] - 현재 블록의 칼라 컴포넌트를 규정하는 변수 cIdx
- [0686] - 현재 블록의 예측 샘플들을 규정하는 (nCurrSw)x(nCurrSh) 배열 predSamples,
- [0687] - 현재 블록의 잔차 샘플(residual sample)들을 규정하는 (nCurrSw)x(nCurrSh) 배열 resSamples.
- [0688] 이 프로세스의 출력은 재생 화면 샘플 배열 recSamples과 IBC 버퍼 배열 ibcBuf이다.
- [0689] 칼라 컴포넌트(cIdx)의 값에 따라 다음 할당들이 이뤄지는데:
- [0690] - cIdx가 0이면, recSamples가 재생 화면 샘플 배열( $S_L$ )에 대응하고 함수 clipCidx1가 Clip1<sub>y</sub>에 대응한다.
- [0691] - 그렇지 않고 cIdx가 1이면, tuCbfChroma가 tu\_cbf\_cb[xCurr][yCurr]로 설정되고, recSamples가 재생 크로마 샘플 배열( $S_{cb}$ )에 대응하고 함수 clipCidx1가 Clip1<sub>c</sub>에 대응한다.
- [0692] - 그렇지 않으면(cIdx가 2), tuCbfChroma가 tu\_cbf\_cr[xCurr][yCurr]로 설정되고, recSamples가 재생 크로마 샘플 배열( $S_{cr}$ )에 대응되고 함수 clipCidx1가 Clip1<sub>c</sub>에 대응된다.
- [0693] slice\_lmcs\_enabled\_flag의 값에 따라, 다음이 적용된다:
- [0694] - slice\_lmcs\_enabled\_flag이 0이면 위치(xCurr, yCurr)에서의 재생 샘플들의 (nCurrSw)x(nCurrSh) 블록이  $i = 0..nCurrSw - 1, j = 0..nCurrSh - 1$ 에 대해 다음과 같이 도출된다:
- [0695]  $recSamples[xCurr + i][yCurr + j] = clipCidx1(predSamples[i][j] + resSamples[i][j])$  (8-992)
- [0696] - 그렇지 않으면(slice\_lmcs\_enabled\_flag이 1), 다음이 적용된다:
- [0697] - cIdx가 0이면 다음이 적용된다:
- [0698] - 8.7.5.2항에 규정된 루마 샘플들에 대한 매핑 프로세스에 의한 화면 재생이 루마 위치(xCurr, yCurr)와, 블록 폭 nCurrSw 및 높이 nCurrSh와, 예측 루마 샘플 행렬 predSamples와, 및 잔여 루마 샘플 행렬 resSamples를 입력으로 호출되고, 출력은 재생 루마 샘플 배열 recSamples이다.
- [0699] - 그렇지 않으면(cIdx가 0보다 크면) 8.7.5.3항에 규정된 크로마 샘플들에 대한 루마 의존 크로마 잔여 스케일링 프로세스(luma dependent chroma residual scaling process)가 크로마 위치(xCurr, yCurr)와, 폭 nCurrSw 및 높이 nCurrSh를 가지는 변환 블록과, 현재 크로마 변환 블록의 코딩된 블록 플렉 tuCbfChroma와, 예측 크로마 샘플 배열 predSamples와, 및 잔여 크로마 샘플 배열 resSamples를 입력으로 호출되고, 그 출력은 재생 크로마 샘플 배열 recSamples이다.
- [0700] 현재 코딩 유닛의 디코딩 후, 다음이 적용된다:
- [0701]  $i = 0..nCurrSw - 1, j = 0..nCurrSh - 1$ 에 대해
- [0702]  $ibcBuf[(xCurr + i) \% wIbcBuf][(yCurr + j) \% ctbSize] = recSamples[xCurr + i][yCurr + j]$
- [0703] **5.18 실시예 #18(Embodiment #18)**
- [0704] 이 명세서에서 일부 예들에서의 변경은 굵고 밑줄 친 이탤릭체 텍스트로 표시된다.
- [0705] **7.3.7 슬라이스 데이터 선택스(Slice data syntax)**

[0706] 7.3.7.1 일반적 슬라이스 데이터 선택스(General slice data syntax)

표 8

| slice_data() {                                                                          | Descriptor |
|-----------------------------------------------------------------------------------------|------------|
| for(i = 0; i < NumBricksInCurrSlice; i++) {                                             |            |
| CtbAddrInBs = FirstCtbAddrBs[SliceBrickIdx[i]]                                          |            |
| for(j = 0; j < NumCtusInBrick[SliceBrickIdx[i]]; j++, CtbAddrInBs++) {                  |            |
| if(((j % BrickWidth[SliceBrickIdx[i]]) == 0) {                                          |            |
| NumHmvpCand = 0                                                                         |            |
| NumHmvpIbcCand = 0                                                                      |            |
| resetIbcBuf = 1                                                                         |            |
| }                                                                                       |            |
| CtbAddrInRs = CtbAddrBsToRs[CtbAddrInBs]                                                |            |
| coding_tree_unit()                                                                      |            |
| if(entropy_coding_sync_enabled_flag && ((j + 1) % BrickWidth[SliceBrickIdx[i]] == 0)) { |            |
| end_of_subset_one_bit /* equal to 1 */                                                  | ae(v)      |
| if(j < NumCtusInBrick[SliceBrickIdx[i]] - 1)                                            |            |
| byte_alignment()                                                                        |            |
| }                                                                                       |            |
| }                                                                                       |            |
| if(!entropy_coding_sync_enabled_flag) {                                                 |            |
| end_of_brick_one_bit /* equal to 1 */                                                   | ae(v)      |
| if(i < NumBricksInCurrSlice - 1)                                                        |            |
| byte_alignment()                                                                        |            |
| }                                                                                       |            |
| }                                                                                       |            |

[0708] 7.4.8.5 코딩 유닛 시맨틱들(Coding unit semantics)

[0709] 이하의 모든 조건들이 참이면, 공유된 병합 후보 목록 영역에 대한 사용이력 기반 모션 벡터 예측변수 목록이 NumHmvpSmrIbcCand를 NumHmvpIbcCand로 설정하고,  $i = 0..NumHmvpIbcCand - 1$ 에 대해 HmvpSmrIbcCandList[i]를 HmvpIbcCandList[i]로 설정함으로써 갱신된다:

[0710] - IsInSmr[x0][y0]이 TRUE이다.

[0711] - SmrX[x0][y0]이 x0이다.

[0712] - SmrY[x0][y0]이 y0이다.

[0713]  $x = x0..x0 + cbWidth - 1$  and  $y = y0..y0 + cbHeight - 1$ 에 대해 다음 할당들이 이뤄진다:

[0714]  $CbPosX[x][y] = x0$  (7-135)

[0715]  $CbPosY[x][y] = y0$  (7-136)

[0716]  $CbWidth[x][y] = cbWidth$  (7-137)

[0717]  $CbHeight[x][y] = cbHeight$  (7-138)

[0718]  $rSize$ 를  $\min(ctbSize, 64)$ 로  $wIbcBufY$ 를  $(128*128/CtbSizeY)$ 로 설정한다.

[0719]  $ibcBuf_{\text{L}}$ 는  $wIbcBufY$ 인 폭과  $CtbSizeY$ 인 높이를 갖는 배열이다.

[0720]  $ibcBuf_{\text{Cb}}$ 와  $ibcBuf_{\text{Cr}}$ 은  $wIbcBufC = (wIbcBufY/SubWidthC)$ 인 폭과  $(CtbSizeY/SubHeightC)$ 인 높이, 즉  $CtbSizeC$ 를 갖는 배열들이다.

[0721]  $resetIbcBuf$ 가 1이면 다음이 적용된다

- [0722]  $- \underline{x = x0..x0 + wIbcBufY - 1}$  및  $\underline{y = y0..y0 + CtbSizeY - 1}$ 에 대해  $\underline{ibcBuf_{cb}[x \% wIbcBufY][y \% CtbSizeY] = -1}$
- [0723]  $- \underline{x = x0..x0 + wIbcBufC - 1}$  and  $\underline{y = y0..y0 + CtbSizeC - 1}$ 에 대해  $\underline{ibcBuf_{cb}[x \% wIbcBufC][y \% CtbSizeC] = -1}$
- [0724]  $- \underline{x = x0..x0 + wIbcBufC - 1}$  and  $\underline{y = y0..y0 + CtbSizeC - 1}$ 에 대해  $\underline{ibcBuf_{cr}[x \% wIbcBufC][y \% CtbSizeC] = -1}$
- [0725]  $- \underline{resetIbcBuf = 0}$
- [0726]  $\underline{(x0 \% vSizeY)}$ 가 0이고  $\underline{(y0 \% vSizeY)}$ 가 0이면 다음이 적용된다
- [0727]  $- \underline{ibcBuf_{cb}[x \% wIbcBufY][y \% CtbSizeY] = -1}$ , for  $\underline{x = x0..x0 + vSize - 1}$  and  $\underline{y = y0..y0 + vSize - 1}$
- [0728]  $- \underline{x = x0/SubWidthC..x0/SubWidthC + vSize/SubWidthC - 1}$  및  $\underline{y = y0/SubHeightC..y0/SubHeightC + vSize/SubHeightC - 1}$ 에 대해  $\underline{ibcBuf_{cb}[x \% wIbcBufC][y \% CtbSizeC] = -1}$
- [0729]  $- \underline{x = x0/SubWidthC..x0/SubWidthC + vSize/SubWidthC - 1}$  및  $\underline{y = y0/SubHeightC..y0/SubHeightC + vSize/SubHeightC - 1}$ 에 대해  $\underline{ibcBuf_{cr}[x \% wIbcBufC][y \% CtbSizeC] = -1}$
- [0730] **8.6.2 IBC 블록들에 대한 모션 벡터의 도출 프로세스(Derivation process for motion vector components for IBC blocks)**
- [0731] **8.6.2.1 개요(General)**
- [0732] 이 프로세스에 대한 입력들은:
- [0733] - 현재 화면의 상-좌 루마 샘플에 대한 현재 루마 코딩 블록의 상-좌 샘플의 루마 위치(xCb, yCb),
- [0734] - 루마 샘플들 내의 현재 코딩 블록의 폭을 규정하는 변수 cbWidth,
- [0735] - 루마 샘플들 내의 현재 코딩 블록의 높이를 규정하는 변수 cbHeight.
- [0736] 이 프로세스의 출력들은:
- [0737] - 1/16 프랙셔널 샘플 정확도의 루마 모션 벡터(mvL).
- [0738] 루마 모션 벡터(mvL)는 다음과 같이 도출된다:
- [0739] - 8.6.2.2항에 규정된 IBC 루마 모션 벡터 예측변수의 도출 프로세스가 루마 위치(xCb, yCb)와, 변수 cbWidth 및 cbHeight 입력들로 호출되고, 출력은 루마 모션 벡터(mvL)이다.
- [0740] - general\_merge\_flag[xCb][yCb]가 0이면 다음이 적용된다:
- [0741] 4. 변수 mvd가 다음과 같이 도출된다:
- [0742]  $\underline{mvd[0] = MvdL0[xCb][yCb][0]}$  (8-883)
- [0743]  $\underline{mvd[1] = MvdL0[xCb][yCb][1]}$  (8-884)
- [0744] 5. 8.5.2.14항에 규정된 모션 벡터들에 대한 반올림 프로세스가 mvL로 설정된 mvX와, MvShift + 2,로 설정된 rightShift와, 및 MvShift + 2로 설정된 leftShift를 입력으로 호출되고, 출력은 반올림된 mvL이다.

6. 루마 모션 벡터(mvL)가 다음과 같이 수정된다:

$$u[0] = (mvL[0] + mvd[0] + 2^{18}) \% 2^{18} \quad (8-885)$$

$$mvL[0] = (u[0] \geq 2^{17}) ? (u[0] - 2^{18}) : u[0] \quad (8-886)$$

$$u[1] = (mvL[1] + mvd[1] + 2^{18}) \% 2^{18} \quad (8-887)$$

$$mvL[1] = (u[1] \geq 2^{17}) ? (u[1] - 2^{18}) : u[1] \quad (8-888)$$

**주목** 1 - 위에 규정된 mvL[0] 및 mvL[1]의 결과 값들은 항상 양단을 포함하여  $-2^{17}$  내지  $2^{17} - 1$ 의 범위 내에 있을 것이다.

8.6.2.6항에 규정된 사용-이력 기반 모션 벡터 예측변수에 대한 갱신 프로세스가 루마 모션 벡터(mvL)을 입력으로(with) 호출된다.

8.6.2.5항은 mvL을 입력으로 호출되어 mvC를 출력한다.

루마 블록 벡터(mvL)가 다음 제약들을 준수하는 것이 비트스트림 적합성의 요건이다:

-  $((yCb + (mvL[1] \gg 4)) \% CtbSizeY) + cbHeight$ 가  $CtbSizeY$  이하이다

-  $x = xCb..xCb + cbWidth - 1$  및  $y = yCb..yCb + cbHeight - 1$ 에 대해,  $ibcBuf_{cb}[(x + (mvL[0] \gg 4)) \% wIbcBufY][(y + (mvL[1] \gg 4)) \% CtbSizeY]$ 가 -1이 아니어야 한다.

-  $x = xCb..xCb + cbWidth - 1$  및  $y = yCb..yCb + cbHeight - 1$ 에 대해  $treeType$ 이  $SINGLE\_TREE$ 이며,  $ibcBuf_{cb}[(x + (mvC[0] \gg 5)) \% wIbcBufC][(y + (mvC[1] \gg 5)) \% CtbSizeC]$ 가 -1이 아니어야 한다.

### 8.6.3 IBC 블록들에 대한 디코딩 프로세스(Decoding process for ibc blocks)

#### 8.6.3.1 개요(General)

이 프로세스는 IBC 예측 모드로 디코딩된 코딩 유닛을 디코딩할 때 호출된다.

이 프로세스에 대한 입력들은:

- 현재 화면의 상-좌 루마 샘플에 대한 현재 코딩 블록의 상-좌 샘플을 규정하는 루마 위치(xCb, yCb),

- 루마 샘플들 내의 현재 코딩 블록의 폭을 규정하는 변수 cbWidth,

- 루마 샘플들 내의 현재 코딩 블록의 높이를 규정하는 변수 cbHeight,

- 현재 블록의 칼라 구성요소 인덱스.

- 모션 벡터 mv.

-  $(wIbcBufY)_x(CtbSizeY)$  배열  $ibcBuf_{cb}$ ,  $(wIbcBufC)_x(CtbSizeC)$  배열  $ibcBuf_{cb}$ ,  $(wIbcBufC)_x(CtbSizeC)$  배열  $ibcBuf_{cr}$ .

이 프로세스의 출력들은:

- 예측 샘플들의 배열 predSamples.

$x = xCb..xCb + Width - 1$  and  $y = yCb..yCb + Height - 1$ 에 대해, 다음이 적용된다

$cIdx$ 가 0이면

$predSamples[x][y] = ibcBuf_{cb}[(x + mv[0] \gg 4)) \% wIbcBufY][(y + (mv[1] \gg 4)) \% CtbSizeY]$

- [0769] cIdx가 1이면
- [0770] 
$$\text{predSamples}[x][y] = \text{ibcBuf}_{cb}[(x + mv[0] \gg 5)) \% wIbcBufC][(y + (mv[1] \gg 5)) \% CtbSizeC]$$
- [0771] cIdx가 2이면
- [0772] 
$$\text{predSamples}[x][y] = \text{ibcBuf}_{cr}[(x + mv[0] \gg 5)) \% wIbcBufC][(y + (mv[1] \gg 5)) \% CtbSizeC]$$
- [0773] **8.7.5 화면 재생 프로세스(Picture reconstruction process)**
- [0774] **8.7.5.1 개요(General)**
- [0775] 이 프로세스에 대한 입력들은:
- [0776] - 현재 화면 컴포넌트의 상-좌 샘플에 대한 현재 블록의 상-좌 샘플을 규정하는 위치(xCurr, yCurr),
- [0777] - 각각 현재 블록의 폭 및 높이를 규정하는 변수 nCurrSw 및 nCurrSh ,
- [0778] - 현재 블록의 칼라 컴포넌트를 규정하는 변수 cIdx,
- [0779] - 현재 블록의 예측 샘플들을 규정하는 (nCurrSw)x(nCurrSh) 배열 predSamples,
- [0780] - 현재 블록의 잔차 샘플들을 규정하는 (nCurrSw)x(nCurrSh) 배열 resSamplesk.
- [0781] 이 프로세스의 출력들은 재생 화면 샘플 배열 recSamples과 IBC buffer arrays ibcBuf<sub>l</sub>, ibcBuf<sub>cb</sub>, ibcBuf<sub>cr</sub>이다.
- [0782] 칼라 컴포넌트 cIdx의 값에 따라 다음 할당이 이뤄진다:
- [0783] - cIdx가 0이면, recSamples가 재생 화면 샘플 배열(S<sub>L</sub>)에 대응하고 함수 clipCidx1이 Clip1<sub>y</sub>에 대응한다.
- [0784] - 그렇지 않고 cIdx가 1이면, tuCbfcChroma가 tu\_cbf\_cb[xCurr][yCurr]로 설정되고, recSamples가 재생 크로마 샘플 배열(S<sub>cb</sub>)에 대응하며 함수 clipCidx1이 Clip1<sub>c</sub>에 대응한다.
- [0785] - 그렇지 않으면(cIdx가 2이면), tuCbfcChroma가 tu\_cbf\_cr[xCurr][yCurr]으로 설정되고, recSamples가 재생 크로마 샘플 배열(S<sub>cr</sub>)에 대응하며, 함수 clipCidx1이 Clip1<sub>c</sub>에 대응한다.
- [0786] slice\_lmcs\_enabled\_flag의 값에 따라 다음이 적용된다:
- [0787] - slice\_lmcs\_enabled\_flag이 0이면, 위치(xCurr, yCurr)에서의 재생 샘플(recSamples)들의 (nCurrSw)x(nCurrSh) 블록이 i = 0..nCurrSw - 1, j = 0..nCurrSh - 1에 대해 다음과 같이 호출된다:
- [0788] 
$$\text{recSamples}[xCurr + i][yCurr + j] = \text{clipCidx1}(\text{predSamples}[i][j] + \text{resSamples}[i][j]) \quad (8-992)$$
- [0789] - 그렇지 않으면(slice\_lmcs\_enabled\_flag가 1이면), 다음이 적용된다:
- [0790] - cIdx가 0이면, 다음이 적용된다:
- [0791] - 8.7.5.2항에 규정된 루마 샘플들에 대한 매핑 프로세스에 의한 화면 재생이 루마 위치(xCurr, yCurr)와, 블록 폭 nCurrSw 및 높이 nCurrSh와, 예측 루마 샘플 배열 predSamples와, 및 잔여 루마 샘플 배열 resSamples들을 입력으로 호출되고, 그 출력은 재생 루마 샘플 배열 recSamples이다.
- [0792] - 그렇지 않으면(cIdx가 0보다 크면), 8.7.5.3항에 규정된 크로마 샘플들에 대한 루마 의존 크로마 잔여 스케일링 프로세스에 의한 화면 재생이 크로마 위치(xCurr, yCurr)와, 변환 블록 폭 nCurrSw 및 높이 nCurrSh와, 현재 크로마 변환 블록의 코딩된 블록 플렉 tuCbfcChroma와, 예측 크로마 샘플 배열 predSamples와, 및 잔여 크로마 샘플 배열 resSamples들을 입력으로 호출되고, 그 출력은 재생 루마 샘플 배열 recSamples이다.
- [0793] 현재 코딩 유닛의 디코딩 후, 다음이 적용될 수 있다:
- [0794] cIdx가 0이고, treeType이 SINGLE\_TREE 또는 DUAL\_TREE\_LUMA이면, 다음이 적용된다

[0795]  $i = 0..nCurrSw - 1, j = 0..nCurrSh - 1$ 에 대해

[0796]  $ibcBuf_l[(xCurr + i) \% wIbcBufY][(yCurr + j) \% CtbSizeY] = recSamples[xCurr + i][yCurr + j].$

[0797]  $cIdx$ 가 1이고,  $treeType$ 이 SINGLE TREE 또는 DUAL TREE CHROMA이면, 다음이 적용된다

[0798]  $i = 0..nCurrSw - 1, j = 0..nCurrSh - 1$ 에 대해

[0799]  $ibcBuf_{cb}[(xCurr + i) \% wIbcBufC][(yCurr + j) \% CtbSizeC] = recSamples[xCurr + i][yCurr + j].$

[0800]  $cIdx$ 가 2이고,  $treeType$ 이 SINGLE TREE 또는 DUAL TREE CHROMA이면, 다음이 적용된다

[0801]  $i = 0..nCurrSw - 1, j = 0..nCurrSh - 1$ 에 대해

[0802]  $ibcBuf_{cr}[(xCurr + i) \% wIbcBufC][(yCurr + j) \% CtbSizeC] = recSamples[xCurr + i][yCurr + j].$

[0803] 5.19 실시예 #19(Embodiment #19)

[0804] 이 명세서에서 일부 예들에서의 변경은 굵고 밑줄 친 이탤릭체 텍스트로 표시된다.

[0805] 7.3.7 슬라이스 데이터 선택스(Slice data syntax)

[0806] 7.3.7.1 일반적 슬라이스 데이터 선택스(General slice data syntax)

Ⅴ 9

| [0807] | slice_data() {                                                         | Descriptor |
|--------|------------------------------------------------------------------------|------------|
|        | for(i = 0; i < NumBricksInCurrSlice; i++) {                            |            |
|        | CtbAddrInBs = FirstCtbAddrBs[SliceBrickIdx[i]]                         |            |
|        | for(j = 0; j < NumCtusInBrick[SliceBrickIdx[i]]; j++, CtbAddrInBs++) { |            |
|        | if((j % BrickWidth[SliceBrickIdx[i]] == 0) {                           |            |
|        | NumHmvpCand = 0                                                        |            |
|        | NumHmvpIbcCand = 0                                                     |            |
|        | <u>resetIbcBuf = 1</u>                                                 |            |
|        | }                                                                      |            |
|        | CtbAddrInRs = CtbAddrBsToRs[CtbAddrInBs]                               |            |
|        | coding_tree_unit()                                                     |            |
|        | if(entropy_coding_sync_enabled_flag &&                                 |            |
|        | ((j + 1) % BrickWidth[SliceBrickIdx[i]] == 0)) {                       |            |
|        | <b>end_of_subset_one_bit</b> /* equal to 1 */                          | ae(v)      |
|        | if(j < NumCtusInBrick[SliceBrickIdx[i]] - 1)                           |            |
|        | byte_alignment()                                                       |            |
|        | }                                                                      |            |
|        | }                                                                      |            |
|        | if(!entropy_coding_sync_enabled_flag) {                                |            |
|        | <b>end_of_brick_one_bit</b> /* equal to 1 */                           | ae(v)      |
|        | if(i < NumBricksInCurrSlice - 1)                                       |            |
|        | byte_alignment()                                                       |            |
|        | }                                                                      |            |
|        | }                                                                      |            |
|        | }                                                                      |            |

[0808] 7.4.8.5 코딩 유닛 시맨틱들(Coding unit semantics)

[0809] 다음 모든 조건들이 참이면, 공유된 병합 후보 목록 영역에 대한 사용이력 기반 모션 벡터 예측변수가, NumHmvpSmrIbcCand를 NumHmvpIbcCand와 함께 설정하고,  $i = 0 \dots \text{NumHmvpIbcCand} - 1$ 에 대해 HmvpSmrIbcCandList[i]를 HmvpIbcCandList[i]와 함께 함으로써 갱신된다.:

- [0810] - IsInSmr[x0][y0]이 TRUE이다.
- [0811] - SmrX[x0][y0]가 x0이다.

- [0812] -  $\text{SmrY}[x0][y0]$ 가  $y0$ 이다.
- [0813]  $x = x0..x0 + \text{cbWidth} - 1$  및  $y = y0..y0 + \text{cbHeight} - 1$ 에 대해 다음 할당들이 이뤄진다:
- [0814]  $\text{CbPosX}[x][y] = x0$  (7-135)
- [0815]  $\text{CbPosY}[x][y] = y0$  (7-136)
- [0816]  $\text{CbWidth}[x][y] = \text{cbWidth}$  (7-137)
- [0817]  $\text{CbHeight}[x][y] = \text{cbHeight}$  (7-138)
- [0818]  $vSize$ 를  $\min(\text{ctbSize}, 64)$ 으로  $wIbcBufY$ 를  $(128*128/\text{ctbSizeY})$ 로 설정한다.
- [0819]  $i\text{bcBuf}_i$ 는  $wIbcBufY$ 인 폭과  $\text{CtbSizeY}$ 인 높이를 갖는 배열이다.
- [0820]  $i\text{bcBuf}_{\text{Cb}}$  및  $i\text{bcBuf}_{\text{Cr}}$ 들은  $wIbcBufC = (wIbcBufY/\text{SubWidthC})$ 인 폭과  $(\text{CtbSizeY}/\text{SubHeightC})$ 인 높이, 즉  $\text{CtbSizeC}$ 를 갖는 배열이다.
- [0821]  $\text{resetIbcBuf}$ 가 1이면 다음이 적용된다
- [0822] -  $x = x0..x0 + wIbcBufY - 1$  및  $y = y0..y0 + \text{CtbSizeY} - 1$ 에 대해  $i\text{bcBuf}_i[x \% wIbcBufY][y \% \text{CtbSizeY}] = -1$
- [0823] -  $x = x0..x0 + wIbcBufC - 1$  및  $y = y0..y0 + \text{CtbSizeC} - 1$ 에 대해  $i\text{bcBuf}_{\text{Cb}}[x \% wIbcBufC][y \% \text{CtbSizeC}] = -1$
- [0824] -  $x = x0..x0 + wIbcBufC - 1$  및  $y = y0..y0 + \text{CtbSizeC} - 1$ 에 대해  $i\text{bcBuf}_{\text{Cr}}[x \% wIbcBufC][y \% \text{CtbSizeC}] = -1$
- [0825] -  $\text{resetIbcBuf} = 0$
- [0826]  $(x0 \% vSizeY)$ 가 0이고  $(y0 \% vSizeY)$ 가 0이면 다음이 적용된다
- [0827] -  $x = x0..x0 + \min(vSize, \text{cbWidth}) - 1$  및  $y = y0..y0 + \min(vSize, \text{cbHeight}) - 1$ 에 대해  $i\text{bcBuf}_i[x \% wIbcBufY][y \% \text{CtbSizeY}] = -1$
- [0828] -  $x = x0/\text{SubWidthC}..x0/\text{SubWidthC} + \min(vSize/\text{SubWidthC}, \text{cbWidth}) - 1$  및  $y = y0/\text{SubHeightC}..y0/\text{SubHeightC} + \min(vSize/\text{SubHeightC}, \text{cbHeight}) - 1$ 에 대해  $i\text{bcBuf}_{\text{Cb}}[x \% wIbcBufC][y \% \text{CtbSizeC}] = -1$
- [0829] -  $x = x0/\text{SubWidthC}..x0/\text{SubWidthC} + \min(vSize/\text{SubWidthC}, \text{cbWidth}) - 1$  및  $y = y0/\text{SubHeightC}..y0/\text{SubHeightC} + \min(vSize/\text{SubHeightC}, \text{cbHeight}) - 1$ 에 대해  $i\text{bcBuf}_{\text{Cr}}[x \% wIbcBufC][y \% \text{CtbSizeC}] = -1$
- [0830] **8.6.2 IBC 블록들에 대한 모션 벡터 컴포넌트의 도출 프로세스(Derivation process for motion vector components for IBC blocks)**
- [0831] **8.6.2.1 개요(General)**
- [0832] 이 프로세스에 대한 입력들은:
- [0833] - 현재 화면의 상-좌 루마 샘플에 대한 현재 루마 코딩 블록의 상-좌 샘플의 루마 위치( $x_{\text{Cb}}, y_{\text{Cb}}$ ),
- [0834] - 루마 샘플들 내의 현재 코딩 블록의 폭을 규정하는 변수  $\text{cbWidth}$ ,
- [0835] - 루마 샘플들 내의 현재 코딩 블록의 높이를 규정하는 변수  $\text{cbHeight}$ .
- [0836] 이 프로세스의 출력들은:
- [0837] - 1/16 프랙셔널 샘플 정확도의 루마 모션 벡터( $\text{mvL}$ ).
- [0838] 이 루마 모션 벡터( $\text{mvL}$ )는 다음과 같이 도출된다:

- [0839] - 8.6.2.2항에 규정된 IBC 루마 모션 벡터 예측에 대한 도출 프로세스가 루마 위치(xCb, yCb), 변수 cbWidth 및 cbHeight들을 입력으로 호출되고 그 출력은 루마 모션 벡터(mvL)이다.
- [0840] - general\_merge\_flag[xCb][yCb]가 0이면 다음이 적용된다:
- [0841] 7. 변수 mvd가 다음과 같이 도출된다:
- [0842]  $mvd[0] = MvdL0[xCb][yCb][0]$  (8-883)
- [0843]  $mvd[1] = MvdL0[xCb][yCb][1]$  (8-884)
- [0844] 8. 8.5.2.14항에 규정된 모션 벡터에 대한 반올림 프로세스가 입력들로 mvX를 mvL로, rightShift를 MvShift + 2로, 그리고 leftShift를 MvShift + 2로 설정하여 호출되고 반올림된 mvL을 출력으로 한다.
- [0845] 9. 루마 모션 벡터 mvL가 다음과 같이 수정된다:
- $u[0] = (mvL[0] + mvd[0] + 2^{18}) \% 2^{18}$  (8-885)
- $mvL[0] = (u[0] \geq 2^{17}) ? (u[0] - 2^{18}) : u[0]$  (8-886)
- $u[1] = (mvL[1] + mvd[1] + 2^{18}) \% 2^{18}$  (8-887)
- $mvL[1] = (u[1] \geq 2^{17}) ? (u[1] - 2^{18}) : u[1]$  (8-888)
- [0846]
- [0847] **주목** 1 - 위에 규정된 mvL[0] 및 mvL[1]의 결과 값들은 항상 양단을 포함하여  $-2^{17}$  내지  $2^{17} - 1$ 의 범위 내에 있을 것이다.
- [0848] 8.6.2.6항에 규정된 사용이력 기반 모션 벡터 예측변수에 대한 갱신 프로세스가 루마 모션 벡터 mvL을 입력으로 (with) 호출된다.
- [0849] 8.6.2.5항은 mvL을 입력으로 mvC를 출력으로 호출된다.
- [0850] 루마 블록 벡터 mbl이 다음 제약들을 준수해야한다는 것이 비트스트림 적합성의 요건이다:
- [0851] -  $((yCb + (mvL[1] \gg 4)) \% CtbSizeY) + cbHeight) \geq CtbSizeY$ 이다.
- [0852] -  $x = xCb..xCb + cbWidth - 1$  및  $y = yCb..yCb + cbHeight - 1$ 에 대해,  $ibcBuf_{cb}[(x + (mvL[0] \gg 4)) \% wIbcBufY][(y + (mvL[1] \gg 4)) \% CtbSizeY]$ 가 -1이 아니어야 한다.
- [0853] - treeType이 SINGLE\_TREE이면,  $x = xCb..xCb + cbWidth - 1$  및  $y = yCb..yCb + cbHeight - 1$ 에 대해,  $ibcBuf_{cb}[(x + (mvC[0] \gg 5)) \% wIbcBufC][(y + (mvC[1] \gg 5)) \% CtbSizeC]$ 가 -1이 아니어야 한다.
- [0854] **8.6.3 IBC 블록들에 대한 디코딩 프로세스(Decoding process for ibc blocks)**
- [0855] **8.6.3.1 개요(General)**
- [0856] 이 프로세스는 IBC 예측 모드로 코딩된 코딩 유닛을 디코딩할 때 호출된다.
- [0857] 이 프로세스의 입력들은:
- [0858] - 현재 화면의 상-좌 샘플에 대한 현재 코딩 블록의 상-좌 샘플을 규정하는 루마 위치(xCb, yCb),
- [0859] - 루마 샘플들 내의 현재 코딩 블록의 폭을 규정하는 변수 cbWidth,
- [0860] - 루마 샘플들 내의 현재 코딩 블록의 높이를 규정하는 변수 cbHeight,
- [0861] - 현재 블록의 칼라 구성요소 인덱스를 규정하는 변수 cIdx.
- [0862] - 모션 벡터 mv.
- [0863] -  $(wIbcBufY)x(CtbSizeY)$  배열  $ibcBuf_{cb}$ ,  $(wIbcBufC)x(CtbSizeC)$  배열  $ibcBuf_{cb}$ ,  $(wIbcBufC)x(CtbSizeC)$  배열

$ibcBuf_{Cr}$ .

이 프로세스의 출력들은:

- 예측 샘플들의 배열  $predSamples$ .

$x = xCb..xCb + Width - 1$  and  $y = yCb..yCb + Height - 1$ 에 대해, 다음이 적용된다

$cIdx$ 가 0이면

$predSamples[x][y] = ibcBuf_Y[(x + mv[0] >> 4)) \% wIbcBufY][(y + (mv[1] >> 4)) \% CtbSizeY]$

$cIdx$ 가 1이면

$predSamples[x][y] = ibcBuf_Cb[(x + mv[0] >> 5)) \% wIbcBufC][(y + (mv[1] >> 5)) \% CtbSizeC]$

$cIdx$ 가 2이면

$predSamples[x][y] = ibcBuf_{Cr}[(x + mv[0] >> 5)) \% wIbcBufC][(y + (mv[1] >> 5)) \% CtbSizeC]$

## 8.7.5 화면 재생 프로세스(Picture reconstruction process)

### 8.7.5.1 개요(General)

이 프로세스에 대한 입력들은:

- 현재 화면 컴포넌트의 상-좌 샘플에 대한 현재 블록의 상-좌 샘플을 규정하는 위치( $xCurr$ ,  $yCurr$ ),

- 각각 현재 블록의 폭 및 높이를 규정하는 변수  $nCurrSw$  및  $nCurrSh$ ,

- 현재 블록의 칼라 컴포넌트를 규정하는 변수  $cIdx$ ,

- 현재 블록의 예측 샘플들을 규정하는  $(nCurrSw) \times (nCurrSh)$  배열  $predSamples$ ,

- 현재 블록의 잔차 샘플들을 규정하는  $(nCurrSw) \times (nCurrSh)$  배열  $resSamples$ .

이 프로세스의 출력은 재생 화면 샘플 배열  $recSamples$ 과 IBC 버퍼 배열  $ibcBuf_L$ ,  $ibcBuf_Cb$ ,  $ibcBuf_{Cr}$ 들이다

칼라 컴포넌트  $cIdx$ 의 값에 따라, 다음 할당들이 이뤄진다:

-  $cIdx$ 가 0이면,  $recSamples$ 가 재생 화면 샘플 배열( $S_L$ )에 대응하고 함수  $clipCidx1$ 이  $Clip1_Y$ 에 대응한다.

- 그렇지 않고  $cIdx$ 가 1이면,  $tuCbfChroma$ 가  $tu_cbf_{cb}[xCurr][yCurr]$ 로 설정되고,  $recSamples$ 가 재생 크로마 샘플 배열( $S_{Cb}$ )에 대응하며 함수  $clipCidx1$ 이  $Clip1_C$ 에 대응한다.

- 그렇지 않으면( $cIdx$ 가 2이면),  $tuCbfChroma$ 가  $tu_cbf_{cr}[xCurr][yCurr]$ 로 설정되고,  $recSamples$ 가 재생 크로마 샘플 배열( $S_{Cr}$ )에 대응하고 함수  $clipCidx1$ 이  $Clip1_C$ 에 대응한다.

$slice\_lmcs\_enabled\_flag$ 의 값에 따라 다음이 적용된다:

-  $slice\_lmcs\_enabled\_flag$ 이 0이면, 위치( $xCurr$ ,  $yCurr$ )에서의 재생 샘플들  $recSamples$ 의  $(nCurrSw) \times (nCurrSh)$  블록이  $i = 0..nCurrSw - 1$ ,  $j = 0..nCurrSh - 1$ 에 대해 다음과 같이 도출된다:

$recSamples[xCurr + i][yCurr + j] = clipCidx1(predSamples[i][j] + resSamples[i][j])$  (8-992)

- 그렇지 않으면( $slice\_lmcs\_enabled\_flag$ 이 1이면), 다음이 적용된다:

-  $cIdx$ 가 0이면 다음이 적용된다:

- 8.7.5.2항에 규정된 루마 샘플들에 대한 매핑 프로세스에 의한 화면 재생이 루마 위치( $xCurr$ ,  $yCurr$ )와, 블록 폭  $nCurrSw$  및 높이  $nCurrSh$ 와, 예측 루마 샘플 배열  $predSamples$ , 및 잔여 루마 샘플 배열  $resSamples$ 들을 입력으로 호출되고, 그 출력은 재생 루마 샘플 배열  $recSamples$ 이다.

[0892] - 그렇지 않으면( $cIdx$ 가 0보다 크면), 8.7.5.3항에 규정된 크로마 샘플들에 대한 루마 의존 크로마 잔여 스케일링 프로세스에 의한 화면 재생이 크로마 위치( $xCurr$ ,  $yCurr$ )와, 변환 블록 폭  $nCurrSw$  및 높이  $nCurrSh$ 와, 현재 크로마 변환 블록의 코딩된 블록 플렉  $tuCbfChroma$ 와, 예측 크로마 샘플 배열  $predSamples$ 와, 및 잔여 크로마 샘플 배열  $resSamples$ 들을 입력으로 호출되고, 그 출력은 재생 크로마 샘플 배열  $recSamples$ 이다.

[0893] 현재 코딩 유닛의 디코딩 후, 다음이 적용될 수 있다:

[0894]  $cIdx$ 가 0이고,  $treeType$ 이  $SINGLE\_TREE$  또는  $DUAL\_TREE\_LUMA$ 이면, 다음이 적용된다

[0895]  $i = 0..nCurrSw - 1$ ,  $j = 0..nCurrSh - 1$ 에 대해

[0896]  $IBCBuf_{cb}[(xCurr + i) \% wIbcBufY][(yCurr + j) \% CtbSizeY] = recSamples[xCurr + i][yCurr + j]$ .

[0897]  $cIdx$ 가 1이고,  $treeType$ 이  $SINGLE\_TREE$  또는  $DUAL\_TREE\_CHROMA$ 이면, 다음이 적용된다

[0898]  $i = 0..nCurrSw - 1$ ,  $j = 0..nCurrSh - 1$ 에 대해

[0899]  $IBCBuf_{cb}[(xCurr + i) \% wIbcBufC][(yCurr + j) \% CtbSizeC] = recSamples[xCurr + i][yCurr + j]$ .

[0900]  $cIdx$ 가 2이고  $treeType$ 이  $SINGLE\_TREE$  또는  $DUAL\_TREE\_CHROMA$ 이면, 다음이 적용된다

[0901]  $i = 0..nCurrSw - 1$ ,  $j = 0..nCurrSh - 1$ 에 대해

[0902]  $IBCBuf_{cr}[(xCurr + i) \% wIbcBufC][(yCurr + j) \% CtbSizeC] = recSamples[xCurr + i][yCurr + j]$ .

[0903] 5.20 실시예 #20(Embodiment #20)

[0904] 이 명세서에서 일부 실시들에서의 변경은 굵고 밑줄 친 이탤릭 체 텍스트로 표시된다.

[0905] 7.3.7 슬라이스 데이터 선택스(Slice data syntax)

[0906] 7.3.7.1 일반적 슬라이스 데이터 선택스(General slice data syntax)

표 10

| slice_data() {                                                                                         | Descriptor |
|--------------------------------------------------------------------------------------------------------|------------|
| for( $i = 0$ ; $i < NumBricksInCurrSlice$ ; $i++$ ) {                                                  |            |
| $CtbAddrInBs = FirstCtbAddrBs[SliceBrickIdx[i]]$                                                       |            |
| for( $j = 0$ ; $j < NumCtusInBrick[SliceBrickIdx[i]]$ ; $j++$ , $CtbAddrInBs++$ ) {                    |            |
| if( $((j \% BrickWidth[SliceBrickIdx[i]]) == 0)$ {                                                     |            |
| $NumHmvpCand = 0$                                                                                      |            |
| $NumHmvpIbcCand = 0$                                                                                   |            |
| $resetIbcBuf = 1$                                                                                      |            |
| }                                                                                                      |            |
| $CtbAddrInRs = CtbAddrBsToRs[CtbAddrInBs]$                                                             |            |
| coding_tree_unit()                                                                                     |            |
| if( $(entropy\_coding\_sync\_enabled\_flag \ \&\& \ ((j + 1) \% BrickWidth[SliceBrickIdx[i]] == 0))$ { |            |
| end_of_subset_one_bit /* equal to 1 */                                                                 | ae(v)      |
| if( $j < NumCtusInBrick[SliceBrickIdx[i]] - 1$ )                                                       |            |
| byte_alignment()                                                                                       |            |
| }                                                                                                      |            |
| }                                                                                                      |            |
| if(! $entropy\_coding\_sync\_enabled\_flag$ ) {                                                        |            |
| end_of_brick_one_bit /* equal to 1 */                                                                  | ae(v)      |
| if( $i < NumBricksInCurrSlice - 1$ )                                                                   |            |
| byte_alignment()                                                                                       |            |
| }                                                                                                      |            |
| }                                                                                                      |            |

[0908] **7.4.8.5 코딩 유닛 시맨틱들(Coding unit semantics)**

[0909] 다음 모든 조건들이 참이면, 공유 병합 후보 목록 영역에 대한 사용이력 기반 모션 벡터 예측변수 목록이 NumHmvpSmrIbcCand를 NumHmvpIbcCand로 설정하고,  $i = 0..NumHmvpIbcCand - 1$ 에 대해 HmvpSmrIbcCandList[i]가 HmvpIbcCandList[i]와 함께 설정함으로써 갱신된다.:

[0910] - IsInSmr[x0][y0]이 TRUE이다.

[0911] - SmrX[x0][y0]이 x0이다.

[0912] - SmrY[x0][y0]이 y0이다.

[0913]  $x = x0..x0 + cbWidth - 1$  및  $y = y0..y0 + cbHeight - 1$ 에 대해 다음 할당들이 이뤄진다:

[0914]  $CbPosX[x][y] = x0$  (7-135)

[0915]  $CbPosY[x][y] = y0$  (7-136)

[0916]  $CbWidth[x][y] = cbWidth$  (7-137)

[0917]  $CbHeight[x][y] = cbHeight$  (7-138)

[0918] vSize를 min(ctbSize, 64)로, wIbcBufY를 (128\*128/CtbSizeY)로 설정한다.

[0919] ibcBuf<sub>l</sub>는 wIbcBufY인 폭과 CtbSizeY인 높이를 가지는 배열이다.

[0920] ibcBuf<sub>cb</sub> 및 ibcBuf<sub>cr</sub>들은 wIbcBufC=(wIbcBufY/SubWidthC)인 폭과 (CtbSizeY/SubHeightC)인 높이를, 즉 CtbSizeC를 갖는 배열들이다.

[0921] resetIbcBuf가 1이면 다음이 적용된다

[0922] -  $x = x0..x0 + wIbcBufY - 1$  및  $y = y0..y0 + CtbSizeY - 1$ 에 대해,  $ibcBuf_l[x \% wIbcBufY][y \% CtbSizeY] = -1$

[0923] -  $x = x0..x0 + wIbcBufC - 1$  및  $y = y0..y0 + CtbSizeC - 1$ 에 대해,  $ibcBuf_{cb}[x \% wIbcBufC][y \% CtbSizeC] = -1$

[0924] -  $x = x0..x0 + wIbcBufC - 1$  및  $y = y0..y0 + CtbSizeC - 1$ 에 대해,  $ibcBuf_{cr}[x \% wIbcBufC][y \% CtbSizeC] = -1$

[0925] - resetIbcBuf = 0

[0926] (x0 % vSizeY)가 0이고 (y0 % vSizeY)가 0이면, 다음이 적용된다

[0927] -  $x = x0..x0 + \max(vSize, cbWidth) - 1$  및  $y = y0..y0 + \max(vSize, cbHeight) - 1$ 에 대해,  $ibcBuf_l[x \% wIbcBufY][y \% CtbSizeY] = -1$

[0928] -  $x = x0/SubWidthC..x0/SubWidthC + \max(vSize/SubWidthC, cbWidth) - 1$  및  $y = y0/SubHeightC..y0/SubHeightC + \max(vSize/SubHeightC, cbHeight) - 1$ 에 대해,  $ibcBuf_{cb}[x \% wIbcBufC][y \% CtbSizeC] = -1$

[0929] -  $x = x0/SubWidthC..x0/SubWidthC + \max(vSize/SubWidthC, cbWidth) - 1$  및  $y = y0/SubHeightC..y0/SubHeightC + \max(vSize/SubHeightC, cbHeight) - 1$ 에 대해,  $ibcBuf_{cr}[x \% wIbcBufC][y \% CtbSizeC] = -1$

[0930] **8.6.2 IBC 블록들에 대한 모션 벡터 컴포넌트들의 도출 프로세스(Derivation process for motion vector components for IBC blocks)**

[0931] **8.6.2.1 개요(General)**

[0932] 이 프로세스에 대한 입력들은:

- [0933] - 현재 화면의 상-좌 루마 샘플에 대한 현재 루마 코딩 블록의 상-좌 샘플의 루마 위치(xCb, yCb),
- [0934] - 루마 샘플들 내의 현재 코딩 블록의 폭을 규정하는 변수 cbWidth,
- [0935] - 루마 샘플들 내의 현재 코딩 블록의 높이를 규정하는 변수 cbHeight.
- [0936] 이 프로세스의 출력들은:
- [0937] - 1/16 프랙셔널 샘플 정확도의 루마 모션 벡터 mvL.
- [0938] 루마 모션 벡터 mvL는 다음과 같이 도출된다:
- [0939] - 8.6.2.2 항에 규정된 IBC 루마 모션 벡터 예측의 도출 프로세스가 루마 위치(xCb, yCb), 변수 cbWidth 및 cbHeight들을 입력으로 호출되고, 그 출력은 루마 모션 벡터 mvL이다.
- [0940] - general\_merge\_flag[xCb][yCb]가 0이면, 다음이 적용된다:
- [0941] 10. 변수 mvd가 다음과 같이 도출된다:
- [0942]  $mvd[0] = MvdL0[xCb][yCb][0]$  (8-883)
- [0943]  $mvd[1] = MvdL0[xCb][yCb][1]$  (8-884)
- [0944] 11. 8.5.2.14에 규정된 모션 벡터들의 반올림 프로세스가 입력들로 mvX를 mvL로 설정하고, rightShift를 MvShift + 2로 설정하며, 그리고 leftShift를 MvShift + 2로 설정하여 호출되고, 반올림된 mvL을 출력한다.
- [0945] 12. 루마 모션 벡터 mvL가 다음과 같이 수정된다:
- [0946] 
$$u[0] = (mvL[0] + mvd[0] + 2^{18}) \% 2^{18} \quad (8-885)$$
- [0947] 
$$mvL[0] = (u[0] \geq 2^{17}) ? (u[0] - 2^{18}) : u[0] \quad (8-886)$$
- [0948] 
$$u[1] = (mvL[1] + mvd[1] + 2^{18}) \% 2^{18} \quad (8-887)$$
- [0949] 
$$mvL[1] = (u[1] \geq 2^{17}) ? (u[1] - 2^{18}) : u[1] \quad (8-888)$$
- [0947] 주목 1 - 위에 규정된 mvL[0] 및 mvL[1]의 결과 값들은 항상 양단을 포함하여  $-2^{17}$  내지  $2^{17} - 1$ 의 범위 내에 있을 것이다.
- [0948] 8.6.2.6항에 규정된 사용이력 기반 모션 벡터 예측변수 목록의 갱신 프로세스가 루마 모션 벡터 mvL을 입력으로 (with) 호출된다.
- [0949] 8.6.2.5항이 mvL을 입력으로 mvC를 출력으로 호출된다.
- [0950] 루마 블록 벡터 mvL이 다음 제약들을 준수해야 한다는 것이 비트스트림 적합성의 요건이다:
- [0951] -  $((yCb + (mvL[1] >> 4)) \% CtbSizeY) + cbHeight$ 가  $CtbSizeY$  이하이다.
- [0952] -  $x = xCb..xCb + cbWidth - 1$  및  $y = yCb..yCb + cbHeight - 1$ 에 대해,  $ibcBuf1[(x + (mvL[0] >> 4)) \% wIbcBufY][(y + (mvL[1] >> 4)) \% CtbSizeY]$ 가 -1이어서는 안 된다.
- [0953] 8.6.3 IBC 블록들에 대한 디코딩 프로세스(Decoding process for ibc blocks)
- [0954] 8.6.3.1 개요(General)
- [0955] 이 프로세스는 IBC 예측 모드로 코딩된 코딩 유닛을 디코딩할 때 호출된다.
- [0956] 이 프로세스에 대한 입력들은:
- [0957] - 현재 화면의 상-좌 루마 샘플에 대한 현재 코딩 블록의 상-좌 샘플을 규정하는 루마 위치(xCb, yCb),

- [0958] - 루마 샘플들 내의 코딩 블록의 폭을 규정하는 변수  $cbWidth$ ,
- [0959] - 루마 샘플들 내의 코딩 블록의 높이를 규정하는 변수  $cbHeight$ ,
- [0960] - 현재 블록의 칼라 컴포넌트 인덱스를 규정하는 변수  $cIdx$ .
- [0961] - 모션 벡터  $mv$ .
- [0962] -  $(wIbcBufY)_x(CtbSizeY)$  배열  $ibcBuf_{l_{-}}$ ,  $(wIbcBufC)_x(CtbSizeC)$  배열  $ibcBuf_{cb_{-}}$ ,  $(wIbcBufC)_x(CtbSizeC)$  배열  $ibcBuf_{cr_{-}}$ .
- [0963] 이 프로세스의 출력들은:
- [0964] - 예측 샘플들의 배열  $predSamples$ .
- [0965]  $x = xCb..xCb + Width - 1$  및  $y = yCb..yCb + Height - 1$ 에 대해, 다음이 적용된다
- [0966]  $cIdx$ 가 0이면
- [0967]  $predSamples[x][y] = ibcBuf_{l_{-}}[(x + mv[0] >> 4)] \% wIbcBufY[(y + (mv[1] >> 4))] \% CtbSizeY]$
- [0968]  $cIdx$ 가 1이면
- [0969]  $predSamples[x][y] = ibcBuf_{cb_{-}}[(x + mv[0] >> 5)] \% wIbcBufC[(y + (mv[1] >> 5))] \% CtbSizeC]$
- [0970]  $cIdx$ 가 2이면
- [0971]  $predSamples[x][y] = ibcBuf_{cr_{-}}[(x + mv[0] >> 5)] \% wIbcBufC[(y + (mv[1] >> 5))] \% CtbSizeC]$
- [0972] **8.7.5 화면 재생 프로세스(Picture reconstruction process)**
- [0973] **8.7.5.1 개요(General)**
- [0974] 이 프로세스에 대한 입력들은:
- [0975] - 현재 화면 컴포넌트의 상-좌 샘플에 대한 현재 블록의 상-좌 샘플을 규정하는 위치( $xCurr$ ,  $yCurr$ ),
- [0976] - 각각 현재 블록의 폭 및 높이를 규정하는 변수  $nCurrSw$  및  $nCurrSh$ ,
- [0977] - 현재 블록의 칼라 컴포넌트를 규정하는 변수  $cIdx$ ,
- [0978] - 현재 블록의 예측 샘플들을 규정하는  $(nCurrSw) \times (nCurrSh)$  배열  $predSamples$ ,
- [0979] - 현재 블록의 잔차 샘플들을 규정하는  $(nCurrSw) \times (nCurrSh)$  배열  $resSamples$ .
- [0980] 이 프로세스의 출력은 재생 화면 샘플 배열  $recSamples$ 과 **IBC 버퍼 배열**  $ibcBuf_{l_{-}}$ ,  $ibcBuf_{cb_{-}}$ ,  $ibcBuf_{cr_{-}}$ 들이다.
- [0981] 칼라 컴포넌트  $cIdx$ 의 값에 따라 다음 할당이 이뤄진다:
- [0982] -  $cIdx$ 가 0이면,  $recSamples$ 가 재생 화면 샘플 배열( $S_l$ )에 대응하고 함수  $clipCidx1$ 이  $Clip1_Y$ 에 대응한다.
- [0983] - 그렇지 않고  $cIdx$ 가 1이면,  $tuCbfChroma$ 가  $tu\_cbf\_cb[xCurr][yCurr]$ 로 설정되고,  $recSamples$ 가 재생 크로마 샘플 배열( $S_{cb}$ )에 대응하며 함수  $clipCidx1$ 이  $Clip1_C$ 에 대응한다.
- [0984] - 그렇지 않으면( $cIdx$ 가 2),  $tuCbfChroma$ 가  $tu\_cbf\_cr[xCurr][yCurr]$ 로 설정되고,  $recSamples$ 가 재생 크로마 샘플 배열( $S_{cr}$ )에 대응하며 함수  $clipCidx1$ 이  $Clip1_C$ 에 대응한다.
- [0985]  $slice\_lmcs\_enabled\_flag$ 의 값에 따라 다음이 적용된다:
- [0986] -  $slice\_lmcs\_enabled\_flag$ 이 0이면, 위치( $xCurr$ ,  $yCurr$ )에서의 재생 샘플들  $recSamples$ 의  $(nCurrSw) \times (nCurrSh)$  블록이  $i = 0..nCurrSw - 1$ ,  $j = 0..nCurrSh - 1$ 에 대해 다음과 같이 도출된다:

- [0987]  $recSamples[xCurr + i][yCurr + j] = clipCidx1(predSamples[i][j] + resSamples[i][j])$  (8-992)
- [0988] - 그렇지 않으면(slice\_lmcs\_enabled\_flag이 1이면), 다음이 적용된다:
- [0989] - cIdx가 0이면, 다음이 적용된다:
- [0990] - 8.7.5.2항에 규정된 루마 샘플들에 대한 매핑 프로세스에 의한 화면 재생이 루마 위치(xCurr, yCurr)와, 블록 폭 nCurrSw 및 높이 nCurrSh와, 예측 루마 샘플 배열 predSamples와, 및 잔차 루마 샘플 배열 resSamples들을 입력으로 호출되고, 그 출력은 재생 루마 샘플 배열 recSamples이다.
- [0991] - 그렇지 않으면(cIdx가 0보다 크면), 8.7.5.3항에 규정된 크로마 샘플들에 대한 루마 의존 크로마 잔여 스케일링 프로세스에 의한 화면 재생이 크로마 위치(xCurr, yCurr)와, 변환 블록 폭 nCurrSw 및 높이 nCurrSh와, 현재 크로마 변환 블록의 코딩된 블록 플렉 tuCbFChroma와, 예측 크로마 샘플 배열 predSamples와, 및 잔차 크로마 샘플 배열 resSamples들을 입력으로 호출되고, 그 출력은 재생 크로마 샘플 배열 recSamples이다.
- [0992] 현재 코딩 유닛의 디코딩 후, 다음이 적용될 수 있다:
- [0993] cIdx가 0이고 treeType이 SINGLE\_TREE 또는 DUAL\_TREE\_LUMA이면, 다음이 적용된다
- [0994]  $i = 0..nCurrSw - 1, j = 0..nCurrSh - 1$ 에 대해
- [0995]  $ibcBuf_L[(xCurr + i) \% wIbcBufY][(yCurr + j) \% CtbSizeY] = recSamples[xCurr + i][yCurr + j].$
- [0996] cIdx가 1이고 treeType이 SINGLE\_TREE 또는 DUAL\_TREE\_CHROMA이면, 다음이 적용된다
- [0997]  $i = 0..nCurrSw - 1, j = 0..nCurrSh - 1$ 에 대해
- [0998]  $ibcBuf_{Cb}[(xCurr + i) \% wIbcBufC][(yCurr + j) \% CtbSizeC] = recSamples[xCurr + i][yCurr + j].$
- [0999] cIdx가 2이고 treeType이 SINGLE\_TREE 또는 DUAL\_TREE\_CHROMA이면, 다음이 적용된다
- [1000]  $i = 0..nCurrSw - 1, j = 0..nCurrSh - 1$ 에 대해
- [1001]  $ibcBuf_{Cr}[(xCurr + i) \% wIbcBufC][(yCurr + j) \% CtbSizeC] = recSamples[xCurr + i][yCurr + j].$
- [1002] 도 6은 시각 매체(비디오 또는 화상) 처리의 예시적 방법(600)의 흐름도이다. 방법(600)은 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드를 사용하여 현재 비디오 블록에 대한 참조 샘플들을 저장할 버퍼의 크기의 결정(602)과, 버퍼 내에 저장된 참조 샘플들을 사용하는 변환의 수행(604)을 포함한다.
- [1003] 다음 항들은 방법(600) 및 다른 방법들로 구현된 일부 예시적 바람직한 특징들을 설명한다. 추가적 예들은 이 명세서의 섹션 4에 제공된다.
- [1004] 1. 비디오 처리 방법은: 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드를 사용하여 현재 비디오 블록에 대한 참조 샘플들을 저장할 버퍼의 크기의 결정하는 단계와; 및 버퍼 내에 저장된 참조 샘플들을 사용하여 변환을 수행하는 단계를 포함한다.
- [1005] 2. 1항의 방법에서, 버퍼의 크기는 소정의 상수이다.
- [1006] 3. 1-2항의 방법에서, M과 N이 정수들일 때, 크기는 MxN이다.
- [1007] 4. 3항의 방법에서, MxN은 64x64 또는 128x128 또는 64x128이다.
- [1008] 5. 1항의 방법에서, 버퍼의 크기는 현재 비디오 블록의 코딩 트리 유닛의 크기와 같다.
- [1009] 6. 1항의 방법에서, 버퍼의 크기는 변환 동안 사용되는 가상 파이프라인 데이터 유닛의 크기와 같다.
- [1010] 7. 1항의 방법에서, 버퍼의 크기는 비트스트림 내의 필드(field)에 대응한다.
- [1011] 8. 7항의 방법에서, 필드는 비디오 파라미터 세트(set) 또는 시퀀스 파라미터 세트 또는 화면 파라미터 세트 또는 화면 헤더 또는 슬라이드 헤더 또는 타일 그룹 헤더 레벨의 비트스트림 표현 내에 포함된다.
- [1012] 9. 1-8항 중의 어느 방법에서, 버퍼의 크기는 루마 컴포넌트의 참조 샘플들과 크로마 컴포넌트들의 참조 샘플들

에 대해 다르다.

- [1013] 10. 1-8항 중의 어느 방법에서, 버퍼의 크기는 현재 비디오 블록의 크로마 서브샘플링 포맷(chroma subsampling format)에 좌우된다.
- [1014] 11. 1-8항 중의 어느 방법에서, 참조 샘플들은 RGB 포맷으로 저장된다.
- [1015] 12. 1-11항 중의 어느 방법에서, 버퍼는 루프 필터링 이전과 루프 필터링 이후에 재생 샘플들을 저장하는 데 사용된다.
- [1016] 13. 12항의 방법에서, 루프 필터링이 블록해제(deblocking) 필터링 또는 적응 루프 필터링(adaptive loop filtering; ALF) 또는 샘플 적응 오프셋(sample adaptive offset; SAO) 필터링을 포함한다.
- [1017] 14. 비디오 처리 방법은: 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드를 사용하여 현재 비디오 블록의 참조 샘플들을 저장할 버퍼를 초기화시키는(initializing) 단계와; 및 버퍼 내에 저장된 참조 샘플들의 초기 값들을 사용하여 변환을 수행하는 단계를 포함한다.
- [1018] 15. 14항의 방법에서, 초기 값들은 상수에 대응한다.
- [1019] 16. 14-15항 중의 어느 방법에서, 초기 값들은 현재 비디오 블록의 비트심도의 함수이다.
- [1020] 17. 15항의 방법에서, 상수가 미드 그레이(mid-grey) 값에 해당한다.
- [1021] 18. 14항의 방법에서, 초기 값이 이전에 디코딩된 비디오 블록의 화소값들에 대응한다.
- [1022] 19. 18항의 방법에서, 이전에 디코딩된 비디오 블록이 루프-내(in-loop) 필터링 이전의 디코딩된 블록에 대응한다.
- [1023] 20. 14-19항 중의 어느 방법에서, 버퍼의 크기가 1-13항 중의 하나에 기재된 것이다.
- [1024] 21. 1-20항 중의 어느 방법에서, 버퍼 내의 화소 위치들이  $x$  및  $y$  수들을 사용하여 어드레스된다.
- [1025] 22. 1-20항 중의 어느 방법에서, 버퍼 내의 화소 위치들이 0으로부터  $M*N-1$ 까지 확장되는 단일한 수를 사용하여 어드레스되는데, 여기서  $M$  및  $N$ 은 버퍼의 화소 폭 및 화소 높이이다.
- [1026] 23. 1-20항 중의 어느 방법에서, 현재 비트스트림 표현이 변환을 위한 블록 벡터를 포함하는데, 여기서  $(BV_x, BV_y)$ 로 표기되는 블록 벡터가  $(x-x_0, y-y_0)$ 이고,  $(x_0, y_0)$ 가 현재 비디오 블록의 코딩 트리 유닛의 상-좌 위치에 해당한다.
- [1027] 24. 1-20항 중의 어느 방법에서, 현재 비트스트림 표현이 변환을 위한 블록 벡터를 포함하고,  $(BV_x, BV_y)$ 로 표기되는 이 블록 벡터가  $(x-x_0+Tx, y-y_0+Ty)$ 이고,  $(x_0, y_0)$ 가 현재 비디오 블록의 코딩 트리 유닛의 좌-상 위치에 해당하며,  $T_x$  및  $T_y$ 가 오프셋 값이다.
- [1028] 25. 24항의 방법에서,  $T_x$  및  $T_y$ 가 소정의 오프셋 값이다.
- [1029] 26. 1-20항 중의 어느 방법에서, 변환 동안, 위치  $(x_0, y_0)$ 에 있고 블록 벡터  $(BV_x, BV_y)$ 를 갖는 화소에 대해, 버퍼 내의 대응 기준은 참조 위치  $(x_0+BV_x, y_0+BV_y)$ 에서 찾을 수 있다.
- [1030] 27. 26항의 방법에서, 참조 위치가 버퍼 외부인 경우, 버퍼의 경계에서 클리핑(clipping)됨으로써 기준 내의 버퍼가 결정된다.
- [1031] 28. 26항의 방법에서, 참조 위치가 버퍼 외부인 경우, 기준 내의 버퍼가 소정 값을 갖도록 결정된다.
- [1032] 29. 1-20항 중의 어느 방법에서, 변환 동안, 위치  $(x_0, y_0)$ 에 있고 블록 벡터  $(BV_x, BV_y)$ 를 갖는 화소에 대해, 버퍼 내의 대응 기준은 참조 위치  $((x_0+BV_x) \bmod M, (y_0+BV_y) \bmod N)$ 에서 찾을 수 있는데, 여기서 "mod"는 모듈러 연산이고  $M$  및  $N$ 은 버퍼의  $x$  및  $y$  크기들을 나타내는 정수들이다.
- [1033] 30. 비디오 처리 방법은: 비디오와 현재 비디오 블록 간의 변환 동안 비디오 경계에서 인트라 블록 복사 코딩(모드)을 위한 참조 샘플들을 저장하는 버퍼를 리셋하는 단계와; 및 버퍼에 저장된 참조 샘플들을 사용하여 변환을 수행하는 단계를 포함한다.
- [1034] 31. 30항의 방법에서, 비디오 경계는 새로운 화면 또는 새로운 타일에 대응한다.
- [1035] 32. 30항의 방법에서, 리셋 다음에 버퍼를 가상 파이프라인 데이터 유닛(Virtual Pipeline Data Unit; VPDU)의

재생 값들로 갱신함으로써 변환이 수행된다.

- [1036] 33. 30항의 방법에서, 리셋 다음에 버퍼를 코딩 트리 유닛의 재생 값들로 갱신함으로써 변환이 수행된다.
- [1037] 34. 30항의 방법에서, 리셋이 각 코딩 트리 유닛 행의 시작시 수행된다.
- [1038] 35. 1항의 방법에서, 버퍼의 크기가 L이 정수일 때 L개의 64x64 이전에 디코딩된 블록들에 대응한다.
- [1039] 36. 1-35항 중의 어느 방법에서, 변환 동안 버퍼 내에 샘플들을 독출(reading) 또는 저장하는 데 수직 스캔 순서가 사용된다.
- [1040] 37. 비디오 처리 방법은: 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 코딩 모드를 사용하여 현재 비디오 블록에 대한 참조 샘플들을 저장하는 버퍼를 사용하는 단계로, 버퍼의 제1 비트심도가 코딩된 데이터의 제2 비트심도와 다른 단계와; 및 버퍼에 저장된 참조 샘플들을 사용하여 변환을 수행하는 단계를 포함한다.
- [1041] 38. 37항의 방법에서, 제1 비트심도가 제2 비트심도보다 더 크다.
- [1042] 39. 37-38항 중의 어느 방법에서, 제1 비트심도가 변환 동안 사용되는 재생 버퍼의 비트심도와 동일하다.
- [1043] 40. 37-39항 중의 어느 방법에서, 제1 비트심도가 비트스트림 표현 내에 하나의 값 또는 다른 값으로 신호된다.
- [1044] 41. 37-40 중의 어느 방법에서, 변환이 크로마 및 루마 컴포넌트들에 대해 다른 비트심도들을 사용한다.
- [1045] 37-41항들의 추가적 실시예 및 예들은 섹션 4의 항목 7에 기재되어 있다.
- [1046] 42. 비디오 처리 방법은: 인트라 블록 복사 모드를 사용하여 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환 단계를 포함하는데, 변환 동안의 예측 연산들에 사용되는 제1 정밀도가 변환 동안의 재생 연산들에 사용되는 제2 정밀도보다 더 낮다.
- [1047] 43. 42항의 방법에서, 예측 연산들이  $\text{clip}\{p + [1 \ll (b-1)]\} \gg b, 0, (1 \ll \text{bitdepth}) - 1\} \ll b$ 을 사용하여 재생 샘플 값들로부터 예측 샘플 값을 결정하는데, 여기서 p는 재생 샘플 값이고, b는 소정의 비트 시프트(bit-shifting) 값이며, bitdepth는 예측 샘플 정밀도이다.
- [1048] 42-43항의 추가적인 실시예 및 예들은 섹션 4의 항목 28 내지 31 및 34에 기재되어 있다.
- [1049] 44. 비디오 처리 방법은: 현재 비디오 블록과 인트라 블록 복사 모드를 사용한 현재 비디오 블록의 비트스트림 표현 간의 변환을 수행하는 단계를 구비하는데, 코딩 트리 유닛 크기 MxM에 대해 크기 nMxnM의 기준 블록이 사용되고, 여기서 n과 M은 정수들이며, 현재 비디오 블록이 코딩 트리 유닛 내에 위치하고, 기준 영역이 현재 비디오 블록에 대응하는 코딩 트리 유닛 행 내의 사용 가능한 가장 가까운 n x n 코딩 트리 유닛이다.
- [1050] 44항의 추가적인 실시예 및 예들은 섹션 4의 항목 35에 기재되어 있다.
- [1051] 45. 비디오 처리 방법은: 현재 비디오 블록과 인트라 블록 복사 모드를 사용한 현재 비디오 블록의 비트스트림 표현 간의 변환을 수행하는 단계를 구비하는데, MxM이 아닌 코딩 트리 유닛 크기에 대해 크기 nMxnM의 기준 블록이 사용되고, 여기서 n과 M은 정수들이며, 현재 비디오 블록이 코딩 트리 유닛 내에 위치하고, 기준 영역이 현재 비디오 블록에 대응하는 코딩 트리 유닛 행 내의 사용 가능한 가장 가까운 n x n-1 코딩 트리 유닛이다.
- [1052] 45항의 추가적인 실시예 및 예들은 섹션 4의 항목 36에 기재되어 있다. 도 8 및 9는 이 추가적인 예시적 실시예들을 도시한다.
- [1053] 46. 3항의 방법에서,  $M=mW$  및  $N=H$ 이고, 여기서 W 및 H는 현재 비디오 블록의 코딩 트리 유닛(CTU)의 폭과 높이이며, m은 양의 정수이다.
- [1054] 47. 3항의 방법에서,  $M=W$  및  $N=nH$ 이고, 여기서 W 및 H는 현재 비디오 블록의 코딩 트리 유닛(CTU)의 폭과 높이이며, n은 양의 정수이다.
- [1055] 48. 3항의 방법에서,  $M=mW$  및  $N=nH$ 이고, 여기서 W 및 H는 현재 비디오 블록의 코딩 트리 유닛(CTU)의 폭과 높이이며, m과 n은 양의 정수들이다.
- [1056] 49. 46-48항 중의 어느 방법에서, n과 m은 CTU의 크기에 좌우된다.
- [1057] 50. 비디오 처리 방법은: 비디오의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 비디오의 컴포넌트 X를 사용하여 비디오의 컴포넌트 c의 현재 비디오 블록에 대응하는 블록 벡터의 유효성을 판

단하는 단계로, 컴포넌트 X가 비디오의 루마 컴포넌트와 다른 단계와; 및 블록 벡터가 현재 비디오 블록에 대해 유효하다고 판단하면 이 블록 벡터를 사용하여 변환을 수행하는 단계를 포함한다. 여기서 (BVx, BVy)로 표기되는 블록 벡터는 (x-x0, y-y0)인데, (x0, y0)는 현재 비디오 블록의 코딩 트리 유닛의 상-좌 위치에 해당한다.

- [1058] 51. 50항의 방법에서, 컴포넌트 c는 비디오의 루마 컴포넌트에 대응한다.
- [1059] 52. 50항의 방법에서, 현재 비디오 블록은 크로마 블록이고 비디오는 4:4:4 포맷이다.
- [1060] 53. 50항의 방법에서, 비디오가 4:2:0 포맷이고, 현재 비디오 블록이 (x, y)에서 시작하는 크로마 블록이며, 판단 단계가  $\text{isRec}(c, ((x+Bv_x) \gg 5 < 5) + 64 - (((y+Bv_y) \gg 5) \& 1) * 32 + (x \% 32), ((y+Bv_y) \gg 5 < 5) + (y \% 32))$ 가 참인 경우 블록 벡터가 유효한지 판단하는 단계를 포함한다.
- [1061] 54. 50항의 방법에서, 비디오가 4:2:0 포맷이고, 현재 비디오 블록이 (x, y)에서 시작하는 크로마 블록이며, 판단 단계가  $\text{isRec}(c, x+Bv_x+\text{Chroma\_CTU\_size}, y)$ 가 참인 경우 블록 벡터가 유효한지 판단하는 단계를 포함한다.
- [1062] 55. 비디오 처리 방법은: 비디오 영역의 현재 가상 파이프라인 데이터 유닛(VPDU)의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현을 위해, 비디오 영역의 제1 행으로부터의 K1개의 이전에 처리된 VPDU들과 비디오 영역의 제2 행으로부터의 K2개의 이전에 처리된 VPDU들의 선택적 사용을 결정하는 단계와; 및 변환을 수행하는 단계로, 변환이 현재 VPDU의 나머지 것들을 제외하는 단계를 포함한다.
- [1063] 56. 55항의 방법에서, K1 = 1 및 K2 = 2이다.
- [1064] 57. 55-56항 중의 어느 방법에서, 현재 비디오 블록이 비디오 영역의 크기(dimension) 또는 현재 VPDU의 크기에 기반하여 선택적으로 처리된다.
- [1065] 58. 비디오 처리 방법은: 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 블록 벡터의 유효성 점검을 수행하는 단계로, 블록 벡터가 인트라 블록 복사 모드에 사용되는 단계와; 및 변환 동안 유효성 점검의 결과를 사용하여 블록 벡터를 선택적으로 사용하는 단계를 포함한다.
- [1066] 59. 58항의 방법에서, 인트라 블록 복사(IBC) 버퍼가 변환 동안 사용되는데, 여기서 IBC 버퍼의 폭 및 높이가 Wbuf 및 Hbuf로 표기되고 현재 비디오 블록의 크기가 WxH이며 블록 벡터가 (BVx, BVy)로 표현되고, 현재 비디오 블록이 Wpic 및 Hpic의 크기를 갖는 현재 화면과 Wctu 및 Hctu의 폭과 높이를 갖는 코딩 트리 유닛 내에 위치하며, 유효성 점검은 소정의 규칙(rule)을 사용한다.
- [1067] 60. 58-59항 중의 어느 방법에서, 현재 비디오 블록은 루마 블록, 크로마 블록, 코딩 유닛 CU, 변환 유닛 TU, 4x4 블록, 2x2 블록, 또는 화소 좌표(X, Y)에서 시작하는 모 블록(parent block)의 서브블록이다.
- [1068] 61. 58-60항 중의 어느 방법에서, 유효성 점검 단계가 현재 화면 외부에 속하는(fall) 블록 벡터를 유효한 것으로 간주한다.
- [1069] 62. 58-60항 중의 어느 방법에서, 유효성 점검 단계가 코딩 트리 유닛 외부에 속하는 블록 벡터를 유효한 것으로 간주한다.
- [1070] 이전 섹션의 항목 23-30들은 위 58-62항들의 추가적 예와 변형들을 제공한다.
- [1071] 63. 1-62항 중의 어느 방법에서, 변환 단계가 현재 비디오 블록으로부터 비트스트림 표현을 생성하는 단계를 포함한다.
- [1072] 64. 1-62항 중의 어느 방법에서, 변환 단계가 비트스트림 표현으로부터 현재 비디오 블록의 화소값들을 생성하는 단계를 포함한다.
- [1073] 65. 비디오 인코더 장치는 1-62항들 중의 어떤 하나 이상에 기재된 방법을 구현하도록 구성된 프로세서를 구비한다.
- [1074] 66. 비디오 디코더 장치는 1-62항들 중의 어떤 하나 이상에 기재된 방법을 구현하도록 구성된 프로세서를 구비한다.
- [1075] 67. 저장된 코드를 갖는 컴퓨터 독출 가능한 매체로, 코드는 1-62항들 중의 어떤 하나 이상에 기재된 방법을 구현하는(implementing) 프로세서 실행 가능한 명령들을 구현한다(embodiment).
- [1076] 도 7은 비디오/화상 처리 장치(700)의 하드웨어 플랫폼의 블록도이다. 장치(700)는 이 명세서에 기재된 하나 이상의 방법들을 구현하는 데 사용될 수 있다. 장치(700)는 스마트폰, 태블릿, 컴퓨터, 사물인터넷(Internet of

Things; IoT) 수신기 등 내에 구현될 수 있다. 장치(700)는 하나 이상의 프로세서(702)들과, 하나 이상의 메모리(704)들과, 및 비디오 처리 하드웨어(706)를 포함할 수 있다. 프로세서(들)(702)는 이 명세서에 기재된 (방법 (600)을 제한 없이 포함하는) 하나 이상의 방법들을 구현하도록 구성될 수 있다. 메모리(메모리들)(704)는 이 명세서에 기재된 방법과 기법들을 구현하는 데 사용되는 데이터와 코드를 저장하는 데 사용될 수 있다. 비디오 처리 하드웨어(706)는 하드웨어 회로 내에서 이 명세서에 기재된 일부 기법들을 구현하는 데 사용될 수 있다.

[1077] 현재 비디오 블록에 대응하는 비트스트림 표현은 비트들의 인접한(contiguous) 세트일 필요가 없고 헤더들, 파라미터 세트들, 및 네트워크 추산 계층(network abstraction layer; NAL) 패킷들에 걸쳐 분포될 수 있다.

## [1078] 섹션 A: 다른 추가적인 예시적 실시예

[1079] 섹션 A에서, VVC 표준의 현재 판(version)이 이 명세서에 기재된 기법들의 일부의 구현을 위해 수정될 수 있는 추가적인 예시적 실시예를 제공한다.

[1080] 현재 IBC 기준 버퍼 설계에서의 몇 가지 문제(issue)들을 분석하고 이 문제들을 해결(address)할 다른 설계를 제시한다. 디코딩 메모리와 혼용(mixing)되는 대신 독립적인 IBC 기준 버퍼가 제안된다. 현재의 앵커(current anchor)와 비교하여, 제안된 안(scheme)은 클래스 F에 대해 -0.99%/-0.71%/-0.79% AI/RA/LD-B 루마 BD 비율(BD-rate)과, 6.7% 메모리 절감을 수반하는 4:2:0 TGM에 대해 -2.57%/-1.81%/-1.36%; 또는 클래스 F에 대해 -1.31%/-1.01%/-0.81%과, 6.7% 메모리 증가를 수반하는 4:2:0 TGM에 대해 -3.23%/-2.33%/-1.71%이다.

## [1081] A1. 서론(Introduction)

[1082] 인트라 블록 복사(intra block copy), 즉 IBC(또는 현재 화면 참조(current picture referencing), 즉 이전에는 CPR) 코딩 모드가 채택된다. 이는 IBC 참조 샘플들이 온칩(on-chip) 메모리에 저장되어야 하고, 이에 따라 하나의 CTU의 제한된 기준 영역이 정의됨으로써 구현된다. 버퍼에 대한 추가적(extra) 온칩 메모리를 제한하도록, 현재 설계는 현재 VPDU의 디코딩에 64x64 메모리를 재사용하여 IBC를 지원하는 데 단지 3개의 추가적인 64x64 블록들의 메모리만 필요하다. CU 크기가 128x128일 때, 현재 기준 영역이 도 2에 도시된다.

[1083] 현재 드래프트(VVC 드래프트 4)에서, 영역은 하기와 같이 정의된다.

### 표 11

|        |                                                                                                                                                                                                                                                                                                                                                                                            |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [1084] | - 다음 조건들이 참이어야 한다:                                                                                                                                                                                                                                                                                                                                                                         |
|        | $(yCb + (mvL[1] \gg 4)) \gg CtbLog2SizeY = yCb \gg CtbLog2SizeY$ (8-972)                                                                                                                                                                                                                                                                                                                   |
|        | $(yCb + (mvL[1] \gg 4) + cbHeight - 1) \gg CtbLog2SizeY = yCb \gg CtbLog2SizeY$ (8-973)                                                                                                                                                                                                                                                                                                    |
|        | $(xCb + (mvL[0] \gg 4)) \gg CtbLog2SizeY \geq (xCb \gg CtbLog2SizeY) - 1$ (8-974)                                                                                                                                                                                                                                                                                                          |
|        | $(xCb + (mvL[0] \gg 4) + cbWidth - 1) \gg CtbLog2SizeY \leq (xCb \gg CtbLog2SizeY)$ (8-975)                                                                                                                                                                                                                                                                                                |
|        | [Ed. (SL): 조건 (8-218) 및 (8-216)들은 6.4.X로 점검되었을 수 있다.]                                                                                                                                                                                                                                                                                                                                      |
|        | - $(xCb + (mvL[0] \gg 4)) \gg CtbLog2SizeY$ 가 $(xCb \gg CtbLog2SizeY) - 1$ 이면, 6.4.X항에 규정된 블록 사용 가능성의 도출 프로세스[Ed. (BB): 이웃 블록들 사용 가능성 점검 프로세스 tbd]가 현재 루마 위치(xCurr, yCurr)를 (xCb, yCb)로 설정하고 이웃 루마 위치(((xCb + (mvL[0] >> 4) + CtbSizeY) >> (CtbLog2SizeY - 1)) << (CtbLog2SizeY - 1), ((yCb + (mvL[1] >> 4)) >> (CtbLog2SizeY - 1)) << (CtbLog2SizeY - 1))를 입력으로 호출되는데, 그 출력이 FALSE이어야 한다. |

[1085] 이와 같이, 전체 기준 크기는 CTU이다.

## [1086] A2. 현재 설계의 잠재적 문제들(Potential issues of the current design)

[1087] 현재 설계는 현재 VPDU의 디코딩에 64x64 메모리를 재사용하는 것으로 추정되고, 이에 따라 IBC 기준이 재사용되는 VPDU 메모리에 정렬된다. 이러한 설계는 VPDU 디코딩 메모리를 IBC 버퍼와 한데 묶는다. 몇 가지 문제들이 존재할 수 있는데:

[1088] 1. 더 작은 CTU 크기를 다루는 것이 문제일 수 있다, CTU 크기를 32x32로 가정하면, 현재의 VPDU를 디코딩하기 위한 현재의 64x64 메모리가 다른 아키텍처의 32x32 레벨의 메모리 재사용을 효율적으로 지원할 수 있을 지 여부가 명확하지 않다.

[1089] 2. 기준 영역이 현저히 변동한다. 이에 따라 너무 많은 비트스트림 적합성 제약들이 도입된다. 기준 영역을 효율적으로 이용(exploit)하고 적합한 비트스트림들의 생성을 방지하는 데 인코더에 과도한 부담을 준다. 이는 또한 예를 들어 병합 목록(merge list) 등 다른 모듈들에 비유효한 BV들을 가질 가능성 역시 증가시킨다. 이들 비

유효한 BV들을 처리하기 위해 추가적인 로직들과 추가적인 적합성 제약들을 도입할 수 있다. 이는 인코더 또는 디코더에 부담을 도입할 뿐 아니라, BV 코딩과 MV 코딩 간에 차이(divergence)를 생성할 수도 있다.

[1090] 3. 이 설계는 잘 축소 확대(scale)되지 않는다. VPDU 디코딩이 IBC 버퍼와 혼용되므로, 현재의 단일한(a) 128x128 CTU 설계에 대해 기준 영역을 증가 또는 감소시키기 쉽지 않다. 이는 예를 들어 더 낮거나 높은 프로파일(profile)의 최근의 개발에 있어서 더 우수한 코딩 효율 대(vs.) 온칩 메모리 균형(trade-off)을 이용할 유연성을 제한한다.

[1091] 4. IBC 기준 버퍼의 비트심도가 디코딩 버퍼에 연계된다. 스크린 콘텐츠가 항상 내부 디코딩 비트심도보다 더 낮은 비트심도를 갖더라도, 버퍼는 여전히 주로 반올림 또는 양자화 잡음들을 나타내는 비트들의 저장에 메모리를 소모할 필요가 있다. 이 문제는 더 높은 디코딩 비트심도 구성을 고려할 때 더욱 심해진다.

### [1092] A3. 명확한 IBC 버퍼 설계(A clear IBC buffer design)

[1093] 위 서브섹션에서 열거된 문제들을 해결하기 위해, 디코딩 메모리와 혼용되지 않는 전용(dedicated) IBC 버퍼를 가질 것을 제안한다.

[1094] 128x128 CTU에 대해 버퍼는 8비트 샘플들을 갖는 128x128로 정의되는데, 크기  $w \times h$ 를 갖는 CU( $x, y$ )가 디코딩 되면, 루프 필터링 전의 그 재생이 8비트로 변환되어 위치( $x\%128, y\%128$ )에서 시작되는  $w \times h$  블록 영역에 기입

된다(written). 여기서 모듈러 연산자  $\%$ 는 예를 들어  $x < 0, x \% L \triangleq -(-x \% L)$ 에 대해, 예를 들어  $-3\%128=125$ 의 양수를 회신한다.

[1095] 화소( $x, y$ )가  $BV=(BV_x, BV_y)$ 를 갖고 IBC 모드로 코딩된다고 가정하면, IBC 기준 버퍼에서의 그 예측 샘플은  $((x+BV_x)\%128, (y+BV_y)\%128)$ 에 위치하고 화소값은 예측 전에 10비트로 변환될 것이다.

[1096] 버퍼가 ( $W, H$ )로 고려될 때, ( $x, y$ )에서 시작되는 CTU 또는 CU를 디코딩한 뒤, 루프 필터링 이전의 재생 화소들은  $(x\%W, y\%H)$ 에서 시작되는 버퍼에 저장될 것이다. 이와 같이 CTU를 디코딩한 뒤, 대응 IBC 기준 버퍼가 이에 따라 갱신될 것이다. 이러한 설정은 CTU 크기가 128x128이 아닐 때 발생할 수 있다. 예를 들어, 현재 버퍼 크기를 갖는 64x64 CTU에 대해, 256x64 버퍼가 고려될 수 있다. 64x64 CTU에 대해 도 2가 버퍼 상태를 보인다.

[1097] 도 12는 IBC 기준 버퍼 상태의 도면인데, 여기서 블록은 64x64 CTU를 표기한다.

[1098] 이러한 설계에서, IBC 버퍼가 VPDU 디코딩 메모리와 다르기 때문에, 모든 IBC 기준 버퍼가 기준으로 사용될 수 있다.

[1099] IBC 버퍼의 비트심도가 8비트라면, 3개의 추가적인 10비트 64x64 버퍼를 요구하는 현재 설계에 비해, 온칩 메모리 증가는  $(8*4)/(10*3)-100\%=6.7\%$ 이다.

[1100] 비트심도를 더욱 감소시키면, 메모리 요건이 더욱 감소될 수 있다. 예를 들어, 7비트 버퍼에 대해 온칩메모리 절감(saving)은  $100\%-(7*4)/(10*3)=6.7\%$ 이다.

[1101] 이 설계에 의하면, 유일한(only) 비트스트림 적합성 제약은 기준 블록이 현재 타일의 현재 CTU 행 내의 재생 영역 내부에 위치해야 한다는 것이다.

[1102] 각 CTU 행의 시작시 512로 초기화하는 것이 허용되면, 모든 비트스트림 적합성 제약들이 제거된다.

### [1103] A4. 실험적 결과들(Experimental results)

[1104] 일부 실시예들에서, 본 발명(disclosed) 방법들이 VTM-4.0 소프트웨어를 사용하여 구현될 수 있다.

[1105] 10비트 버퍼 구현과 CTC에 대해, 디코더는 현재의 VTM4.0 인코더와 완전히(fully) 호환적인데, 이는 제안된 디코더가 VTM-4.0 CTC 비트스트림들을 정확히 디코딩한다는 것을 의미한다.

[1106] 7비트 버퍼 구현에 대한 결과가 표 I에 도시되어 있다.

[1107] 8비트 버퍼 구현에 대한 결과가 표 II에 도시되어 있다.

[1108] <표 I. 7 비트 버퍼의 성능. 앵커는 모든 시퀀스들에 IBC를 구현한 VTM-4.0이다.>

표 12

[1109]

|           |                        |        |        |      |      |
|-----------|------------------------|--------|--------|------|------|
|           | All Intra              |        |        |      |      |
|           | Over VTM-4.0 w/ IBC on |        |        |      |      |
|           | Y                      | U      | V      | EncT | DecT |
| Class A1  | -0.01%                 | -0.09% | -0.10% | 132% | 101% |
| Class A2  | 0.05%                  | 0.00%  | 0.06%  | 135% | 100% |
| Class B   | 0.00%                  | -0.02% | 0.01%  | 135% | 100% |
| Class C   | -0.02%                 | 0.01%  | 0.03%  | 130% | 98%  |
| Class E   | -0.13%                 | -0.16% | -0.04% | 135% | 99%  |
| Overall   | -0.02%                 | -0.05% | 0.00%  | 133% | 100% |
| Class D   | 0.04%                  | 0.04%  | 0.12%  | 127% | 107% |
| Class F   | -0.99%                 | -1.14% | -1.18% | 115% | 99%  |
| 4:2:0 TGM | -2.57%                 | -2.73% | -2.67% | 104% | 102% |
|           |                        |        |        |      |      |
|           | Random Access          |        |        |      |      |
|           | Over VTM-4.0 w/ IBC on |        |        |      |      |
|           | Y                      | U      | V      | EncT | DecT |
| Class A1  | 0.02%                  | -0.01% | 0.01%  | 109% | 100% |
| Class A2  | 0.00%                  | -0.04% | 0.03%  | 111% | 100% |
| Class B   | -0.01%                 | -0.10% | -0.22% | 113% | 101% |
| Class C   | -0.01%                 | 0.17%  | 0.12%  | 115% | 100% |
| Class E   |                        |        |        |      |      |
| Overall   | 0.00%                  | 0.00%  | -0.04% | 112% | 100% |
| Class D   | 0.05%                  | 0.16%  | 0.20%  | 117% | 101% |
| Class F   | -0.71%                 | -0.77% | -0.77% | 109% | 99%  |
| 4:2:0 TGM | -1.81%                 | -1.65% | -1.64% | 107% | 101% |
|           |                        |        |        |      |      |
|           | Low delay B            |        |        |      |      |
|           | Over VTM-4.0 w/ IBC on |        |        |      |      |
|           | Y                      | U      | V      | EncT | DecT |
| Class A1  |                        |        |        |      |      |
| Class A2  |                        |        |        |      |      |
| Class B   | 0.01%                  | 0.36%  | 0.30%  | 114% | 95%  |
| Class C   | -0.01%                 | -0.12% | -0.10% | 120% | 98%  |
| Class E   | 0.10%                  | 0.20%  | 0.18%  | 107% | 99%  |
| Overall   | 0.03%                  | 0.16%  | 0.13%  | 114% | 97%  |
| Class D   | -0.01%                 | 1.07%  | 0.18%  | 123% | 104% |
| Class F   | -0.79%                 | -0.89% | -1.01% | 110% | 100% |
| 4:2:0 TGM | -1.36%                 | -1.30% | -1.26% | 109% | 102% |

[1110]

<표 II. 8 비트 버퍼의 성능. 앵커는 모든 시퀀스들에 IBC를 구현한 VTM-4.0이다.>

표 13

[1111]

|          |                        |        |        |      |      |
|----------|------------------------|--------|--------|------|------|
|          | All Intra              |        |        |      |      |
|          | Over VTM-4.0 w/ IBC on |        |        |      |      |
|          | Y                      | U      | V      | EncT | DecT |
| Class A1 | -0.01%                 | 0.02%  | -0.10% | 129% | 102% |
| Class A2 | 0.02%                  | -0.06% | -0.02% | 134% | 102% |
| Class B  | -0.04%                 | -0.02% | -0.07% | 135% | 101% |
| Class C  | -0.03%                 | 0.04%  | 0.00%  | 130% | 98%  |
| Class E  | -0.16%                 | -0.14% | -0.08% | 134% | 100% |
| Overall  | -0.04%                 | -0.03% | -0.05% | 133% | 100% |
| Class D  | 0.00%                  | 0.04%  | 0.02%  | 126% | 101% |
| Class F  | -1.31%                 | -1.27% | -1.29% | 114% | 98%  |

|           |                        |        |        |      |       |
|-----------|------------------------|--------|--------|------|-------|
| 4:2:0 TGM | -3.23%                 | -3.27% | -3.24% | 101% | 100%  |
|           |                        |        |        |      |       |
|           | Random Access          |        |        |      |       |
|           | Over VTM-4.0 w/ IBC on |        |        |      |       |
|           | Y                      | U      | V      | EncT | DecT  |
| Class A1  | -0.01%                 | -0.08% | 0.04%  | 107% | 99%   |
| Class A2  | -0.03%                 | -0.16% | 0.06%  | 110% | 99%   |
| Class B   | -0.01%                 | -0.14% | -0.22% | 111% | 99%   |
| Class C   | -0.01%                 | 0.15%  | 0.09%  | 115% | 100%  |
| Class E   |                        |        |        |      |       |
| Overall   | -0.01%                 | -0.05% | -0.03% | 111% | 99%   |
| Class D   | 0.01%                  | 0.19%  | 0.22%  | 116% | 101%  |
| Class F   | -1.01%                 | -0.99% | -1.01% | 108% | 99%   |
| 4:2:0 TGM | -2.33%                 | -2.14% | -2.19% | 105% | 100%  |
|           |                        |        |        |      |       |
|           | Low delay B            |        |        |      |       |
|           | Over VTM-4.0 w/ IBC on |        |        |      |       |
|           | Y                      | U      | V      | EncT | DecT  |
| Class A1  |                        |        |        |      |       |
| Class A2  |                        |        |        |      |       |
| Class B   | 0.00%                  | 0.04%  | -0.14% | 113% | #NUM! |
| Class C   | -0.05%                 | -0.28% | -0.15% | 119% | 98%   |
| Class E   | 0.04%                  | -0.16% | 0.43%  | 107% | #NUM! |
| Overall   | 0.00%                  | -0.11% | 0.00%  | 113% | #NUM! |
| Class D   | -0.07%                 | 1.14%  | 0.13%  | 122% | 99%   |
| Class F   | -0.81%                 | -0.92% | -0.96% | 111% | 99%   |
| 4:2:0 TGM | -1.71%                 | -1.67% | -1.71% | 106% | 95%   |

[1112] 도 17은 이 명세서에 개시된 다양한 기법들이 구현될 수 있는 예시적 비디오 처리 시스템(1700)을 보이는 블록도이다. 다양한 구현예들은 시스템(1700)의 구성요소들의 일부 또는 전부를 포함할 수 있다. 시스템(1700)은 비디오 콘텐츠(video contents)를 수신하는 입력(1702)을 포함할 수 있다. 비디오 콘텐츠는 예를 들어 8 또는 10 비트 다중 컴포넌트(multi-component) 화소값들 등의 비가공(raw) 또는 비압축 포맷으로 수신되거나 또는 압축 또는 디코딩된 포맷일 수 있다. 입력(1702)은 네트워크 인터페이스, 주변 버스 인터페이스(peripheral bus interface), 또는 저장 인터페이스를 나타낼 수 있다. 네트워크 인터페이스의 예는 Ethernet, 수동형 광 네트워크(passive optical network; PON) 등의 유선 인터페이스와 Wi-Fi 또는 휴대폰 인터페이스(cellular interface) 등의 무선 인터페이스들을 포함한다.

[1113] 시스템(1700)은 이 명세서에 기재된 다양한 코딩 또는 인코딩 방법들을 구현할 수 있는 코딩 구성요소(coding component; 1704)를 포함할 수 있다. 코딩 구성요소(1704)는 입력(1702)으로부터 코딩 구성요소(1704)의 출력으로의 비디오의 평균 비트전송속도(bitrate)를 저하시켜 비디오의 코딩된 표현을 산출할 수 있다. 이에 따라 이 코딩 기법은 때로 비디오 압축 또는 비디오 트랜스코딩(transcoding) 기법으로 지칭된다. 코딩 구성요소(1704)의 출력은 저장되거나 구성 요소(1706)로 표현된 바와 같이 연결된 통신을 통해 전송될 수 있다. 입력(1702)에서 수신된 비디오의 저장 또는 전송된(communicated) 비트스트림(또는 코딩된) 표현은 구성요소(1708)에 사용되어 화소값들 또는 디스플레이 인터페이스(1710)로 전송되는 화상표시 가능한(displayable) 비디오를 생성할 수 있다. 비트스트림으로부터 사용자가 볼 수 있는(user-viewable) 비디오를 생성하는 프로세스는 때로 비디오 압축해제(video decompression)로 지칭된다. 또한 어떤 비디오 처리 작동은 "코딩(coding)" 작동 또는 툴(tool)로 지칭되는데, 코딩 툴 또는 작동이 인코더에서 사용되고, 코딩의 결과를 반전시키는 대응 디코딩 툴 또는 작동이 디코더에서 수행될 것으로 이해해야 할 것이다.

[1114] 주변 버스 인터페이스 또는 디스플레이 인터페이스의 예는 범용 직렬 버스(universal serial bus; USB), 고화질 멀티미디어 인터페이스(high definition multimedia interface; HDMI), 또는 Displayport 등을 포함할 수 있다. 저장 인터페이스의 예는 SATA(serial advanced technology attachment; 직렬 고급 결합 기술; 직렬 ATA), PCI, IDE 인터페이스 등을 포함할 수 있다. 이 명세서에 기재된 기법들은 모바일폰, 랩탑, 스마트폰 또는 디지털 데이터 처리 및/또는 비디오 디스플레이가 가능한 다른 기기들 등의 여러 전자 기기들에 구현될 수

있다.

- [1115] 도 18은 시각 데이터 처리의 예시적 방법의 흐름도이다. 이 흐름도의 단계들은 이 명세서의 섹션 4의 예시적 실시예 18과 연계하여 논의된다. 단계 1802에서, 프로세스는 시각 매체 데이터의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드에서 예측을 위한 참조 샘플들을 저장하는 버퍼를 결정하고, 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관된 움직임 정보에 기초하는 인트라 블록 복사 모드에서 수행된다. 단계 1804에서, 현재 비디오 블록을 포함하고 블록 벡터(BVx, Bvy)를 갖는 코딩 트리 유닛의 좌상단 위치에 대해 현재 비디오 블록의 위치(x0, y0)에 공간적으로 위치한 샘플에 대해, 프로세스는 참조 위치(P, Q)에서의 버퍼에서 대응되는 참조를 산출하고, 참조 위치(P, Q)는 블록 벡터(BVx, Bvy) 및 위치(x0, y0)를 이용하여 결정된다. 단계 1806에서, 프로세스는 참조 위치(P, Q)가 버퍼 외부에 있다고 결정되면, 현재 비디오 블록을 포함하는 코딩 트리 유닛에 대한 현재 비디오 블록의 위치에 적어도 부분적으로 기초하여 참조 위치를 재산출한다.
- [1116] 도 19는 시각 데이터 처리의 예시적 방법의 흐름도이다. 이 흐름도의 단계들은 이 명세서의 섹션 4의 예시적 실시예 29와 연계하여 논의된다. 단계 1902에서, 프로세스는 시각 매체 데이터의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드에서 예측을 위한 참조 샘플들을 저장하는 버퍼를 결정하고, 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관된 움직임 정보에 기반하여 인트라 블록 복사 모드에서 수행된다. 단계 1904에서, 프로세스는 현재 비디오 블록을 포함하고 블록 벡터(BVx, Bvy)를 갖는 코딩 트리 유닛의 좌상단 위치에 대해 현재 비디오 블록의 위치(x0, y0)에 공간적으로 위치한 샘플에 대해, 현재 비디오 블록의 위치(x, y), 현재 비디오 블록의 차원, 화면의 차원, 현재 비디오 블록을 포함하는 코딩 트리 유닛의 차원, 혹은 버퍼의 차원 중 적어도 하나와 연관된 하나 이상의 조건을 만족하는 것에 적어도 부분적으로 기초하여 블록 벡터(Bvx, Bvy)를 할당한다. 단계 1906에서, 프로세스는 블록 벡터(BVx, Bvy)가 유효한지 확인하기 위한 검사를 수행한다. 단계 1908에서, 블록 벡터(BVx, Bvy)가 유효하다고 식별되면, 참조 위치(P, Q)의 버퍼에서 대응되는 참조를 산출하고, 참조 위치(P, Q)를 블록 벡터(BVx, Bvy), 위치(x, y) 및 버퍼의 차원을 이용하여 결정한다.
- [1117] 도 20은 시각 데이터 처리의 예시적 방법의 흐름도이다. 이 흐름도의 단계들은 이 명세서의 섹션 4의 예시적 실시예 19와 연계하여 논의된다. 단계 2002에서, 프로세스는 시각 매체 데이터의 현재 비디오 블록 및 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 현재 비디오 블록에 대한 블록 벡터(BVx, Bvy) 혹은 블록 벡터 차이(BVDx, BVDy)를 결정하고, 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생된 블록과 연관된 움직임 정보에 기초하는 인트라 블록 복사 모드에서 수행된다. 단계 2004에서, 프로세스는 블록 벡터(BVx, Bvy)의 적어도 한 구성요소 혹은 블록 벡터 차이(BVDx, BVDy)의 적어도 한 구성요소가 범위 내에 놓이도록 정규화한다.
- [1118] 도 21은 시각 데이터 처리의 예시적 방법의 흐름도이다. 이 흐름도의 단계들은 이 명세서의 섹션 4의 예시적 실시예 40과 연계하여 논의된다. 단계 2102에서, 프로세스는 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드에서 예측에서 재생 샘플들을 저장하기 위한 버퍼를 결정하고, 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관된 움직임 정보에 기초하는 인트라 블록 복사 모드에서 수행된다. 단계 2104에서, 프로세스는 순서에 따라 버퍼에 저장된 재구성된 샘플들을 업데이트한다.
- [1119] 도 22는 시각 데이터 처리의 예시적 방법의 흐름도이다. 이 흐름도의 단계들은 이 명세서의 섹션 4의 예시적 실시예 57과 연계하여 논의된다. 단계 2202에서 프로세스는 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 수행하고, 변환은 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록에 연관된 움직임 정보에 기반한 인트라 블록 복사 모드에서 수행하되, 변환 동안 예측 산출에 사용되는 제1 정밀도는 재생 산출에 사용되는 제2 정밀도보다 낮다.
- [1120] 도 23은 시각 데이터 처리의 예시적 방법의 흐름도이다. 이 흐름도의 단계들은 이 명세서의 섹션 4의 예시적 실시예 59와 연계하여 논의된다. 단계 2302에서, 프로세스는 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생된 블록과 연관된 움직임 정보에 기초한 인트라 블록 복사 모드를 이용하여 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 수행하되, 변환 동안, nM x nM 크기인 참조 영역이 사용되되, n 및 M은 정수이고 참조 영역은 현재 비디오 블록에 대응되는 코딩 트리 유닛 행에서 가장 가까운 가용한 n x n 코딩 트리 유닛들로부터의 샘플들을 포함한다.
- [1121] 도 24는 시각 데이터 처리의 예시적 방법의 흐름도이다. 이 흐름도의 단계들은 이 명세서의 섹션 4의 예시적 실

시에 60과 연계하여 논의된다. 단계 2402에서, 프로세스는 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관된 움직임 정보에 기초한 인트라 블록 복사 모드를 이용하여 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 수행한다. 변환 동안,  $nM \times pM$  크기인 참조 영역이 사용되며,  $n$ ,  $p$  및  $M$ 은 정수이고 현재 비디오 블록은 코딩 트리 유닛에 위치하고, 참조 영역은 현재 비디오 블록에 대응되는 코딩 트리 유닛 행에서 가장 가까운 가용한  $n \times p-1$  코딩 트리 유닛들로부터의 샘플들을 포함한다.

[1122] 도 25는 시각 데이터 처리의 예시적 방법의 흐름도이다. 이 흐름도의 단계들은 이 명세서의 섹션 4의 예시적 실시예 61과 연계하여 논의된다. 단계 2502에서, 프로세스는 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관된 움직임 정보에 기초하는 인트라 블록 복사 모드를 이용하여 비디오 영역의 VPDU(virtual pipeline data unit)의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 수행하고, 변환 동안  $nM \times nM$  크기인 참조 영역이 사용되고, VPDU의 크기는  $kM \times kM$ 이고, 여기서  $k$ ,  $n$  및  $M$ 은 정수이며 현재 비디오 블록은 코딩 트리 유닛에 위치하고, 참조 영역은 현재 비디오 블록에 대응되는 코딩 트리 유닛 행에서 가장 가까운 가용한  $n \times n-k$  코딩 트리 유닛들로부터 샘플들을 포함한다.

[1123] 도 26은 시각 데이터 처리의 예시적 방법의 흐름도이다. 이 흐름도의 단계들은 이 명세서의 섹션 4의 예시적 실시예 62-66과 연계하여 논의된다. 단계 2602에서, 프로세스는 시각 매체 데이터의  $w \times h$  크기의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드에서 예측을 위한 참조 샘플들을 저장하는 버퍼를 결정하고, 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관된 움직임 정보에 기반하는 인트라 블록 복사 모드에서 수행한다. 단계 2604에서, 프로세스는 현재 비디오 블록을 포함하는  $M \times M$  크기인 코딩 트리 유닛(CTU)의 좌상단 위치에 관련된 현재 비디오 블록의 위치 ( $x_0$ ,  $y_0$ )에 공간적으로 위치한 샘플에 대해, 프로세스는 버퍼에서 참조 위치(P, Q)에서 시작하여 대응되는 참조 영역을 산출한다. 여기서 참조 위치 (P, Q)는 블록 벡터 (BVx, BVy) 및/혹은 위치 ( $x_0$ ,  $y_0$ )를 이용하여 결정된다. 단계 2606에서, 프로세스는 비디오 영역을 갖는 참조 영역의 중복을 제한하기 위해 참조 영역 및/혹은 참조 위치(P, Q)에 하나 이상의 규칙 기반 제한 조건을 적용한다.

[1124] 도 27은 시각 데이터 처리의 예시적 방법의 흐름도이다. 이 흐름도의 단계들은 이 명세서의 섹션 4의 예시적 실시예 68과 연계하여 논의된다. 단계 2702에서, 프로세스는 시각 매체 데이터의 현재 비디오 블록 및 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드에서 예측을 위한 참조 샘플들을 저장하는 버퍼를 결정하고, 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관된 움직임 정보에 기반한 인트라 블록 복사 모드에서 수행한다. 단계 2704에서, 프로세스는 현재 비디오 블록을 포함하는 코딩 유닛(CU)에 연관된 현재 비디오 블록의 위치 ( $x_0$ ,  $y_0$ )에 공간적으로 위치한 샘플에 대해 버퍼에서 참조 위치(P, Q)에서 시작하여 대응되는 참조 영역을 산출한다. 단계 2706에서, 프로세스는 이전에 처리된 블록 중 예측에 사용되는 블록을 결정하기 위해 참조 영역 및 참조 위치를 조정한다.

[1125] 도 28은 시각 데이터 처리의 예시적 방법의 흐름도이다. 이 흐름도의 단계들은 이 명세서의 섹션 4의 예시적 실시예 69-75와 연계하여 논의된다. 단계 2802에서, 프로세스는 비디오의 현재 비디오 블록 및 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 비디오의 구성요소 X를 이용하여 비디오의 구성요소 c의 현재 비디오 블록에 대응되는 블록 벡터의 유효성을 결정하되, 구성요소 X는 비디오의 루마 구성요소와 상이하다. 단계 2804에서, 프로세스는 현재 비디오 블록에 대해 블록 벡터가 유효하다는 결정 하에 블록 벡터를 이용하여 변환을 수행하되, 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관된 움직임 정보에 기반하여 인트라 블록 복사(IBC) 모드에서 수행한다.

[1126] 이제 본 발명의 일부 실시예들이 항 기반 포맷(clause-based format)으로 제공된다.

[1127] A1. 시각 매체 처리 방법은:

[1128] 시각 매체 데이터의 현재 비디오 블록과 상기 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드에서 예측을 위한 참조 샘플들을 저장하는 버퍼를 결정하는 단계, 여기서 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록에 대한 움직임 정보에 기반하는 인트라 블록 복사 모드에서 수행되되;

[1129] 현재 비디오 블록을 포함하고 블록 벡터(BVx, BVy)를 갖는 코딩 트리 유닛의 좌상단 위치에 대한 현재 비디오 블록의 위치 ( $x_0$ ,  $y_0$ )에 공간적으로 위치한 샘플에 대해, 참조 위치(P, Q)에서 버퍼의 대응 참조를 산출하는 단계, 여기서 참조 위치(P, Q)는 블록 벡터(BVx, BVy) 및 위치( $x_0$ ,  $y_0$ )를 이용하여 결정되고;

[1130] 참조 위치(P, Q)가 버퍼 외부에 존재한다는 결정 하에, 현재 비디오 블록을 포함하는 코딩 트리 유닛에 대한 현

재 비디오 블록의 위치에 적어도 부분적으로 기초하여 참조 위치를 재산출하는 단계;를 포함한다.

- [1131] A2. A1항의 방법에 있어서, 참조 위치(P, Q)는  $P=x_0+BV_x$  및  $Q=y_0+BV_y$ 로 결정된다.
- [1132] A3. A1 항의 방법에 있어서, 재산출은,
- [1133] 현재 비디오 블록이 코딩 트리 유닛에 대해 수평적 방향에 위치하는지 혹은 코딩 트리 유닛에 대해 수직적 방향에 위치하는지 여부에 적어도 부분적으로 기초하여 참조 위치(P, Q)를 재산출하는 것을 포함한다.
- [1134] A4. A1 항 내지 A3 항 중 어느 하나 이상의 항에 있어서, 현재 비디오 블록은 상기 코딩 트리 유닛에 대해 수평적 방향에 위치한다.
- [1135] A5. A1 항 내지 A3 항 중 어느 하나 이상의 항에 있어서, 현재 비디오 블록은 코딩 트리 유닛에 대해 수직적 방향에 위치한다.
- [1136] A6. A1항에 있어서, 재산출은
- [1137] 현재 비디오 블록이 시각 매체 데이터의 경계로부터 기 정의된 거리 내에 위치하는지 여부에 적어도 부분적으로 기초하여 참조 위치(P, Q)를 재산출하는 것을 포함한다.
- [1138] A7. A1 항, A2 항 및 A6 항 중 어느 하나 이상의 항에 있어서, 현재 비디오 블록은 시각 매체 데이터의 경계로부터 기 정의된 거리 내에 위치한다.
- [1139] A8. A1항, A2 항 중 어느 하나 이상의 항에 있어서,  $(y_0+BV_y)$ 가 범위  $[0, \dots, N-1]$ 에 존재한다고 결정되면, 참조 위치 (P, Q)에는 기 정의된 값이 할당되며, N은 버퍼의 y 차원을 나타내는 정수이다.
- [1140] A9. A1항, A2 항 중 어느 하나 이상의 항에 있어서,  $(x_0+BV_x)$ 가 범위  $[0, \dots, M-1]$ 에 존재한다고 결정되면, 참조 위치 (P, Q)에는 기 정의된 값이 할당되며, M은 버퍼의 x 차원을 나타내는 정수이다.
- [1141] A10. A1 항, A2 항 중 어느 하나 이상의 항에 있어서, 참조 위치 (P, Q)는  $((x_0+BV_x) \bmod M, y_0+BV_y)$ 로 할당되고, mod는  $x \bmod y = x - y \cdot \text{floor}(x/y)$ 로 정의되는 모듈로(modulo) 연산으로서 floor(a)는 a보다 크지 않은 가장 큰 정수이고 M은 버퍼의 x 차원을 나타내는 정수이다.
- [1142] A11. A1항 내지 A2 항 중 어느 하나 이상의 항에 있어서, 참조 위치 (P, Q)는  $(x_0+BV_x, (y_0+BV_y) \bmod N)$ 로 할당되고, mod는  $x \bmod y = x - y \cdot \text{floor}(x/y)$ 로 정의되는 모듈로(modulo) 연산으로서 floor(a)는 a보다 크지 않은 최대 정수이고 N은 버퍼의 y 차원을 나타내는 정수이다.
- [1143] A12. A10 항에 있어서,  $((x_0+BV_x) \bmod M, y_0+BV_y)$ 가 버퍼 외부라고 결정되면 추가적인 처리가 수행된다.
- [1144] A13. A12 항에 있어서,  $(x_0+BV_x, (y_0+BV_y) \bmod N)$ 가 버퍼 외부라고 결정되면 추가적인 처리가 수행된다.
- [1145] B1. 시각 매체 처리 방법에 있어서,
- [1146] 시각 매체 데이터의 현재 비디오 블록 및 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해, 인트라 블록 복사 모드에서 예측을 위한 참조 샘플들을 저장하는 버퍼를 결정하는 단계, 여기서 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관된 움직임 정보를 기반으로 인트라 블록 복사 모드에서 수행되고;
- [1147] 현재 비디오 블록을 포함하고 블록 벡터( $BV_x, BV_y$ )를 갖는 화면의 좌상단 위치에 대한 현재 비디오 블록의 위치  $(x_0, y_0)$ 에 공간적으로 위치한 샘플에 대해, 현재 비디오 블록의 위치  $(x, y)$ , 현재 비디오 블록의 차원, 화면의 차원, 현재 비디오 블록을 포함하는 코딩 트리 유닛의 차원, 혹은 버퍼의 차원 중 적어도 하나에 연관되는 하나 이상의 조건을 만족하는 것에 적어도 부분적으로 기초하여 블록 벡터 ( $BV_x, BV_y$ )를 유효하다고 할당하는 단계;
- [1148] 블록 벡터( $BV_x, Bvy$ )가 유효한지 확인하기 위한 검사를 수행하는 단계; 및
- [1149] 블록 벡터 ( $BV_x, BV_y$ )가 유효하다고 식별되면, 참조 위치(P, Q)에서 버퍼의 대응 참조를 산출하는 단계;를 포함 하되, 참조 위치(P, Q)는 상기 블록 벡터( $BV_x, BV_y$ ), 위치  $(x, y)$  및 버퍼의 차원을 이용하여 결정된다.
- [1150] B2. B1 항에 있어서, 참조 위치(P, Q)는  $((x+BV_x)\%W_{buf}, (x+BV_y)\%H_{buf})$ 로 결정되고,  $W_{buf} \times H_{buf}$ 는 버퍼의 차원을 지시하고, %는 모듈로 연산을 지시하고,  $x\%y$ 에서  $x < 0$ 이면  $x - y \cdot \text{floor}(x/y)$ 로 정의되고, floor(a)는 a보다 크지 않은 최대 정수이다.

- [1151] C1. 시각 매체 처리 방법에 있어서,
- [1152] 시각 매체 데이터의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 현재 비디오 블록에 대한 블록 벡터 (BVx, BVy) 혹은 블록 벡터 차이 (BVDx, BVDy)를 결정하는 단계, 여기서 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관된 움직임 정보를 기반으로 하는 인트라 블록 복사 모드에서 수행되고,
- [1153] 블록 벡터 (BVx, BVy)의 적어도 한 구성 요소 혹은 블록 벡터 차이 (BVDx, BVDy)의 적어도 한 구성 요소가 범위 내에 놓이도록 정규화하는 단계를 포함한다.
- [1154] C2. C1항의 방법에 있어서, 정규화는
- [1155] 버퍼의 차원을 기반으로 블록 벡터 (BVx, BVy)의 적어도 한 구성 요소 혹은 블록 벡터 차이(BVDx, BVDy)의 적어도 한 구성 요소가 범위 내에 놓이도록 정규화하되, 버퍼는 인트라 블록 복사 모드에서 예측을 위한 참조 샘플들을 저장한다.
- [1156] C3. C2항의 방법에 있어서, 구성요소 BVx는  $(BVx \bmod M)$ 으로 정규화되고, M은 버퍼의 x 차원을 지시하는 정수이 되 "mod" 는  $x \bmod y = x - y \cdot \text{floor}(x/y)$ 로 정의되는 모듈로(modulo) 연산이고, floor(a)는 a보다 크지 않은 최대의 정수이다.
- [1157] C4. C2항의 방법에 있어서, 구성요소 BVx는  $((BVx + x_0) \bmod M) - V$ 로 정규화되되, M은 버퍼의 x 차원을 지시하는 정수이고, "mod" 는  $x \bmod y = x - y \cdot \text{floor}(x/y)$ 로 정의되는 모듈로(modulo) 연산이고, floor(a)는 a보다 크지 않은 최대의 정수이며, V는 기 정의된 값이다.
- [1158] C5. C4항의 방법에 있어서, V는 64이다.
- [1159] C6. C4항의 방법에 있어서, V는  $M/2$ 이다.
- [1160] C7. C4항의 방법에 있어서, V는  $x_0$ 이다.
- [1161] C8. C2항의 방법에 있어서, 구성요소 BVy는  $(BVy \bmod N)$ 로 정규화되고, N은 버퍼의 y 차원을 지시하는 정수이며, "mod" 는  $x \bmod y = x - y \cdot \text{floor}(x/y)$ 로 정의되는 모듈로(modulo) 연산이고, floor(a)는 a보다 크지 않은 최대의 정수이다.
- [1162] C9. C2항의 방법에 있어서, 구성요소 BVy는  $((BVy + y_0) \bmod N) - V$ 로 정규화되고, N은 버퍼의 y차원을 지시하는 정수이며, "mod" 는  $x \bmod y = x - y \cdot \text{floor}(x/y)$ 로 정의되는 모듈로(modulo) 연산이고, V는 기 정의된 값이다.
- [1163] C10. C9항의 방법에 있어서, V는 64이다.
- [1164] C11. C9항의 방법에 있어서, V는  $N/2$ 이다.
- [1165] C12. C9항의 방법에 있어서, V는  $y_0$ 이다.
- [1166] C13. C1항의 방법에 있어서, 구성요소 BVx 및 BVy는 서로 다른 범위 내에 놓이도록 정규화된다.
- [1167] C14. C2항의 방법에 있어서, 구성요소 BVDx는  $(BVDx \bmod M)$ 으로 정규화되고, M은 버퍼의 x차원을 지시하는 정수이며, "mod" 는  $x \bmod y = x - y \cdot \text{floor}(x/y)$ 로 정의되는 모듈로(modulo) 연산이고, floor(a)는 a보다 크지 않은 최대의 정수이다.
- [1168] C15. C2항의 방법에 있어서, 구성요소 BVDx는  $((BVDx + x_0) \bmod M) - V$ 로 정규화되고, M은 버퍼의 x차원을 지시하는 정수이고, "mod" 는  $x \bmod y = x - y \cdot \text{floor}(x/y)$ 로 정의되는 모듈로(modulo) 연산이고, floor(a)는 a보다 크지 않은 최대의 정수이며, V는 기 정의된 값이다.
- [1169] C16. C15항의 방법에 있어서, V는 64이다.
- [1170] C17. C15항의 방법에 있어서, V는  $M/2$ 이다.
- [1171] C18. C15항의 방법에 있어서, V는  $x_0$ 이다.
- [1172] C19. C2항의 방법에 있어서, 구성요소 BVDy는  $(BVDy \bmod N)$ 로 정규화되고, N은 버퍼의 y차원을 지시하는 정수이며, "mod"는 모듈로(modulo) 연산이다.

- [1173] C20. C2항의 방법에 있어서, 구성요소 BVDy는  $((BVDy + y0) \bmod N) - V$ 로 정규화되고, N은 버퍼의 y 차원을 지시하는 정수이며, "mod"는 모듈로(modulo) 연산이고, V는 기 정의된 값이다.
- [1174] C21. C20항의 방법에 있어서, V는 64이다.
- [1175] C22. C20항의 방법에 있어서, V는 N/2이다.
- [1176] C23. C20항의 방법에 있어서, V는 y0이다.
- [1177] C24. C1항의 방법에 있어서, 구성요소 BVDx, BVDy는 서로 다른 범위 내에 놓이도록 정규화된다.
- [1178] D1. 시각 매체 처리 방법에 있어서,
- [1179] 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해, 인트라 블록 복사 모드에서 예측을 위한 재생 샘플들을 저장하는 데 사용되는 버퍼를 결정하는 단계, 여기서 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록과 연관된 움직임 정보를 기반으로 하는 인트라 블록 복사 모드에서 수행되고,
- [1180] 순서에 따라 버퍼에 저장된 재생 샘플들을 업데이트하는 단계를 포함한다.
- [1181] D2. D1항의 방법에 있어서, 버퍼는 순차적인 순서에 따라 업데이트된다.
- [1182] D3. D1 항의 방법에 있어서, 버퍼는 재생 블록의 순서에 따라 업데이트된다.
- [1183] D4. D1항의 방법에 있어서,
- [1184] 버퍼가 가득 찼다는 결정 하에 버퍼에 저장된 재생 샘플들을 최근 재생 샘플들로 대체하는 단계를 더 포함한다.
- [1185] D5. D1항의 방법에 있어서, 버퍼에 저장된 재생 샘플들은 FIFO(First-in-First-out) 순서로 대체된다.
- [1186] D6. D1항의 방법에 있어서, 버퍼에 저장된 재생 샘플들의 가장 오래된 세트가 대체된다.
- [1187] D7. D1항의 방법에 있어서, 버퍼에 저장된 재생 샘플들에 우선순위 값이 할당되고, 버퍼에 저장된 재생 샘플들은 우선순위 값에 따라 대체된다.
- [1188] D8. D1 항의 방법에 있어서, 버퍼에 저장된 재생 샘플들의 서브셋이 향후 대체될 것으로 마킹되며, 서브셋에 포함되지 않은 샘플들은 초기에 대체된다.
- [1189] D9. D7항의 방법에 있어서, 비트스트림 표현에 포함된 플래그는 하나 이상의 조건들을 만족하는 우선순위 값들을 지시한다.
- [1190] D10. D7 항의 방법에 있어서, 우선순위 값들은 현재 비디오 블록의 특성에 기반하여 할당된다.
- [1191] D11. D9 항의 방법에 있어서, 하나 이상의 조건들은 팔레트 모드 및/혹은 인트라 블록 코딩 (IBC) 모드 및/혹은 트랜스폼-스킵(transform-skip) 모드를 이용하여 코딩된 재생 샘플들의 비율에 연관된다.
- [1192] D12. D11항의 방법에 있어서, 팔레트 모드 및/혹은 인트라 블록 코딩 (IBC) 모드 및/혹은 트랜스폼-스킵(transform-skip) 모드를 이용하여 코딩된 재생 샘플들의 비율이 임계값을 초과하였다고 결정되면, 현재 비디오 블록의 모든 샘플들에 높은 우선순위를 할당한다.
- [1193] D13. D12항의 방법에 있어서, 임계값은 현재 비디오 블록의 크기 및/혹은 현재 비디오 블록의 칼라 구성요소 및/혹은 현재 비디오 블록을 포함하는 코딩 트리 유닛(CTU)의 크기에 기반한다.
- [1194] D14. D13항의 방법에 있어서, 임계값은 비트스트림 표현에 필드로서 포함된다.
- [1195] D15. D14항의 방법에 있어서, 필드는 시퀀스 파라미터 세트(PS), 화면 파라미터 세트(PPS), 시퀀스 헤더(sequence header), 슬라이스 헤더(slice header), 타일 그룹(tile group), 타일 레벨(tile level), 혹은 비디오 영역에 포함된다.
- [1196] D16. D4 항의 방법에 있어서, 버퍼에서 가용한 샘플들의 수가 임계값보다 크거나 같다고 결정하는 것은 버퍼가 가득차음을 지시한다.
- [1197] D17. D16 항의 방법에 있어서, 임계값은 64x64x3 루마 샘플이다.
- [1198] E1. 시각 매체 처리 방법에 있어서,
- [1199] 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 수행하는 단계를 포함하되, 변환은 비디

오 블록을 갖는 동일 비디오 영역에 위치한 재생 블록에 연관되는 움직임 정보에 기반하여 인트라 블록 복사 모드에서 수행되고, 변환 동안, 예측 산출에 이용되는 제1 정밀도는 재생 산출에 이용되는 제2 정밀도보다 낮다.

- [1200] E2. E1항의 방법에 있어서, 예측 산출은  $\text{clip}\{p + [1 \ll (b-1)]\} \gg b, 0, (1 \ll \text{bitdepth}) - 1\} \ll b$ 을 이용하여 재생 샘플 값으로부터 예측 샘플 값을 결정하는 것을 포함하되,  $p$ 는 재생 샘플 값이고,  $b$ 는 기 정의된 비트-시프팅(bit-shifting) 값이며,  $\text{bitdepth}$ 는 예측 샘플 정밀도이고, "clip"은  $\text{clip}(x, y, z) = (x < y) ? y : (x > z ? z : x)$ 로 정의되는 클리핑(clipping) 연산이다.
- [1201] E3. E1 항의 방법에 있어서, 예측 산출은  $\text{clip}\{p + [1 \ll (b-1) - 1]\} \gg b, 0, (1 \ll \text{bitdepth}) - 1\} \ll b$ 을 이용하여 재생 샘플 값으로부터 예측 샘플 값을 결정하는 것을 포함하되,  $p$ 는 재생 샘플 값이고,  $b$ 는 기 정의된 기 정의된 비트-시프팅(bit-shifting) 값이며,  $\text{bitdepth}$ 는 예측 샘플 정밀도이고, "clip"은  $\text{clip}(x, y, z) = (x < y) ? y : (x > z ? z : x)$ 로 정의되는 클리핑(clipping) 연산이다.
- [1202] E4. E1항의 방법에 있어서, 예측 산출은  $((p \gg b) + (1 \ll (\text{bitdepth} - 1))) \ll b$ 을 이용하여 재생 샘플 값으로부터 예측 샘플 값을 결정하는 것을 포함하되,  $p$ 는 재생 샘플 값이고,  $b$ 는 기 정의된 비트-시프팅(bit-shifting) 값이며,  $\text{bitdepth}$ 는 예측 샘플 정밀도이다.
- [1203] E5. E1 항의 방법에 있어서, 예측 산출은  $(\text{clip}((p \gg b), 0, (1 \ll (\text{bitdepth} - b))) + (1 \ll (\text{bitdepth} - 1))) \ll b$ 을 이용하여 재생 샘플 값으로부터 예측 샘플 값을 결정하는 것을 포함하되,  $p$ 는 재생 샘플 값이고,  $b$ 는 기 정의된 비트-시프팅(bit-shifting) 값이며,  $\text{bitdepth}$ 는 예측 샘플 정밀도이고, "clip"은  $\text{clip}(x, y, z) = (x < y) ? y : (x > z ? z : x)$ 로 정의되는 클리핑(clipping) 연산이다.
- [1204] E6. E1 항의 방법에 있어서, 예측 산출은 루프 내 정형(in-loop reshaping, ILR) 단계가 적용되는지 여부에 기반한 클리핑 연산을 이용하여 재생 샘플 값으로부터 예측 샘플 값을 결정하는 것을 포함한다.
- [1205] E7. E1 항 내지 E5 항 중 어느 하나 이상의 항의 방법에 있어서,  $b$ 는 재생 샘플과 현재 비디오 블록의 비트심도(bit-depths)의 차이이다.
- [1206] E8. E1항의 방법에 있어서, 제1 정밀도 및/혹은 제2 정밀도 및/혹은 제1 정밀도와 제2 정밀도 간의 차이는 비트 스트림 표현 내의 필드로서 시그널링된다.
- [1207] E9. E1 항의 방법에 있어서, 예측 산출은 재생 샘플 값으로부터 예측 샘플 값을 결정하는 것을 포함하되, 예측 샘플 값의 제1 부분은 제1 정밀도를 갖고, 예측 샘플 값의 제2 부분은 제2 정밀도를 갖는다.
- [1208] E10. E9 항의 방법에 있어서, 현재 비디오 블록은 서로 다른 정밀도를 갖는 샘플들을 포함하는 코딩 트리 유닛 내에 위치한다.
- [1209] E11. E10 항의 방법에 있어서, 서로 다른 정밀도들을 갖는 샘플들은 현재 비디오 블록에 대해 허용된 참조 영역에 포함된다.
- [1210] E12. E9 항의 방법에 있어서, 다른 비디오 블록을 포함하는 코딩 트리 유닛에 연관된 제1 참조 영역은 제1 정밀도를 사용하고 현재 비디오 블록을 포함하는 코딩 트리 유닛과 연관된 제2 참조 영역은 제2 정밀도를 사용한다.
- [1211] E13. E12 항의 방법에 있어서, 제1 참조 영역은 제1 칼라 구성요소에 대응되고, 제2 참조 영역은 제2 칼라 구성요소에 대응된다.
- [1212] F1. 시각 매체 처리 방법에 있어서,
- [1213] 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생된 블록과 연관된 움직임 정보에 기초한 인트라 블록 복사 모드를 이용하여 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 수행하는 단계를 포함하되, 변환 동안,  $nM \times nM$  크기인 참조 영역이 사용되되,  $n$  및  $M$ 은 정수이고 참조 영역은 현재 비디오 블록에 대응되는 코딩 트리 유닛 행에서 가장 가까운 가용한  $n \times n$  코딩 트리 유닛들로부터의 샘플들을 포함한다.
- [1214] F2. F1항의 방법에 있어서, 참조 영역의 크기는  $128 \times 128$  샘플이고, 코딩 트리 유닛 크기는  $64 \times 64$ 이고, 버퍼는 현재 비디오 블록을 포함하는 동일 코딩 트리 유닛 행에서 4개의 가장 가까운 가용한 코딩 트리 유닛들을 포함한다.
- [1215] F3. F1 항의 방법에 있어서, 참조 영역의 크기는  $128 \times 128$  샘플이고, 코딩 트리 유닛의 크기는  $32 \times 32$ 이고, 버퍼는 현재 비디오 블록을 포함하는 동일 코딩 트리 유닛 행에서 16개의 가장 가까운 가용한 코딩 트리 유닛들을 포함한다.

- [1216] G1. 시각 매체 처리 방법에 있어서,
- [1217] 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생된 블록과 연관된 움직임 정보에 기초한 인트라 블록 복사 모드를 이용하여 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 수행하는 단계를 포함하되, 변환 동안,  $nM \times pM$  크기인 참조 영역이 사용되되,  $n$ ,  $p$  및  $M$ 은 정수이고 현재 비디오 블록은 코딩 트리 유닛에 위치하고, 참조 영역은 현재 비디오 블록에 대응되는 코딩 트리 유닛 행에서 가장 가까운 가용한  $n \times p-1$  코딩 트리 유닛들로부터 샘플들을 포함한다.
- [1218] G2. G1항의 방법에 있어서, 참조 영역의 크기는  $128 \times 128$  샘플 혹은  $256 \times 64$  샘플이고, 코딩 트리 유닛의 크기는  $64 \times 64$ 이고, 버퍼는 현재 비디오 블록을 포함하는 동일 코딩 트리 유닛 행에서 가장 가까운 가용한 3개의 코딩 트리 유닛을 포함한다.
- [1219] G3. G1 항의 방법에 있어서, 참조 영역의 크기는  $128 \times 128$  샘플 혹은  $512 \times 32$  샘플이고, 코딩 트리 유닛의 크기는  $32 \times 32$ 이고, 버퍼는 현재 비디오 블록을 포함하는 동일 코딩 트리 유닛 행에서 가장 가까운 가용한 15개의 코딩 트리 유닛을 포함한다.
- [1220] G4. G1항의 방법에 있어서, 코딩 트리 유닛의 크기는  $M \times M$  이다.
- [1221] G5. G1 항 내지 G4 항 중 어느 한 방법에 있어서, 버퍼 외부의 샘플들은 변환 동안 사용이 불허된다.
- [1222] H1. 시각 매체 처리 방법에 있어서,
- [1223] 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생된 블록과 연관된 움직임 정보에 기초하는 인트라 블록 복사 모드를 이용하여 비디오 영역의 VPDU(virtual pipeline data unit)의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 수행하고, 변환 동안  $nM \times nM$  크기인 참조 영역이 사용되고, VPDU의 크기는  $kM \times kM$ 이고, 여기서  $k$ ,  $n$  및  $M$ 은 정수이며 현재 비디오 블록은 코딩 트리 유닛에 위치하고, 참조 영역은 현재 비디오 블록에 대응되는 코딩 트리 유닛 행에서 가장 가까운 가용한  $n \times n-k$  코딩 트리 유닛들로부터의 샘플들을 포함한다.
- [1224] H2. H1항의 방법에 있어서, 참조 영역의 크기는  $128 \times 128$  샘플이고, 코딩 트리 유닛의 크기는  $64 \times 64$ 이고, VPDU의 크기는  $64 \times 64$ 이고, 버퍼는 현재 비디오 블록을 포함하는 동일 코딩 트리 유닛 행에서 가장 가까운 가용한 3개의 코딩 트리 유닛을 포함한다.
- [1225] H3. H1항의 방법에 있어서, 참조 영역의 크기는  $128 \times 128$  샘플이고, 코딩 트리 유닛의 크기는  $32 \times 32$  이고, VPDU의 크기는  $64 \times 64$ 이고, 버퍼는 현재 비디오 블록을 포함하는 동일 코딩 트리 유닛 행에서 가장 가까운 가용한 12개의 코딩 트리 유닛을 포함한다.
- [1226] I1. 시각 매체 처리 방법에 있어서,
- [1227] 시각 매체 데이터의  $w \times h$  크기의 현재 비디오 블록과 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드에서 예측을 위한 참조 샘플들을 저장하는 버퍼를 결정하는 단계, 여기서 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생된 블록과 연관된 움직임 정보에 기반하는 인트라 블록 복사 모드에서 수행하고,
- [1228] 현재 비디오 블록을 포함하는  $M \times M$  크기인 코딩 트리 유닛(CTU)의 좌상단 위치에 관련된 현재 비디오 블록의 위치  $(x_0, y_0)$ 에 공간적으로 위치한 샘플에 대해, 버퍼에서 참조 위치(P, Q)에서 시작하여 대응되는 참조 영역을 블록 벡터 (BVx, BVy) 및/혹은 위치 $(x_0, y_0)$ 을 이용하여 산출하는 단계 및
- [1229] 비디오 영역을 갖는 참조 영역의 중복을 제한하기 위해 참조 영역 및 .혹은 참조 위치(P, Q)에 하나 이상의 규칙 기반 제한 조건을 적용하는 단계를 포함한다.
- [1230] I2. I1 항의 방법에 있어서, CTU 크기가  $128 \times 128$ 이면,  $(x_0, y_0)$ 은  $(m \times 64, n \times 64)$ 로 표현되고, 참조 영역은  $((m-2) \times 64, n \times 64)$ 로 표현되는 좌상 모서리를 갖는 크기  $64 \times 64$ 의 비디오 영역과 겹치지 않도록 제한되며,  $m$  및  $n$ 은 정수이다.
- [1231] I3. I1 항의 방법에 있어서, CTU 크기가  $128 \times 128$ 이면,  $(x_0, y_0)$ 은  $(m \times 64, n \times 64)$ 로 표현되고, 참조 영역은  $(x_0-128, y_0)$ 로 표현되는 좌상 모서리를 갖는 크기  $w \times h$ 의 비디오 영역과 겹치지 않도록 제한되며,  $m$  및  $n$ 은 정수이다.
- [1232] I4. I1 항의 방법에 있어서, 버퍼의 크기는  $kM \times M$ 이고, 참조 영역은  $(x_0-kM, y_0)$ 로 표현되는 좌상 모서리를 갖

는 크기  $w \times h$ 의 비디오 영역과 겹치지 않도록 제한된다.

- [1233] I5. I1 항의 방법에 있어서, 현재 비디오 블록의 좌상 위치가  $(2m*64, 2n*64)$ 로 표현되면, 재생 블록의 좌상 위치가  $((2m-2)*64, 2n*64)$ 로 표현되고,  $m$  및  $n$ 은 정수이고 현재 비디오 블록의 크기는  $64 \times 64$ 이다.
- [1234] I6. I1 항의 방법에 있어서, 현재 비디오 블록의 좌상 위치가  $(2m*64, 2n*64)$ 로 표현되면, 재생 블록의 좌상 위치가  $((2m-1)*64, 2n*64)$ 로 표현되고,  $m$  및  $n$ 은 정수이고 현재 비디오 블록의 크기는  $64 \times 64$ 이다.
- [1235] I7. I1 항의 방법에 있어서, 현재 비디오 블록의 좌상 위치가  $(2m*64, 2n*64)$ 로 표현되면, 재생 블록의 좌상 위치가  $((2m-1)*64, (2n+1)*64)$ 로 표현되고,  $m$  및  $n$ 은 정수이고 현재 비디오 블록의 크기는  $64 \times 64$ 이다.
- [1236] I8. I1 항의 방법에 있어서, 현재 비디오 블록의 좌상 위치가  $(2m*64, 2n*64)$ 로 표현되면, 재생 블록의 좌상 위치는 현재 비디오 블록의 좌상 위치에 위치하고,  $m$  및  $n$ 은 정수이고 현재 비디오 블록의 크기는  $64 \times 64$ 이다.
- [1237] I9. I1 항의 방법에 있어서, 현재 비디오 블록의 좌상 위치가  $((2m+1)*64, (2n+1)*64)$ 로 표현되면, 재생 블록의 좌상 위치는 현재 비디오 블록의 좌상 위치에 위치하고,  $m$  및  $n$ 은 정수이며 현재 비디오 블록의 크기는  $64 \times 64$ 이다.
- [1238] I10. I1 항의 방법에 있어서, 현재 비디오 블록의 좌상 위치가  $((2m+1)*64, 2n*64)$ 로 표현되면, 재생 블록의 좌상 위치는  $((2m-1)*64, 2n*64)$ 로 표현되고,  $m$  및  $n$ 은 정수이며 현재 비디오 블록의 크기는  $64 \times 64$ 이다.
- [1239] I11. I1 항의 방법에 있어서, 현재 비디오 블록의 좌상 위치가  $((2m+1)*64, 2n*64)$ 로 표현되면, 재생 블록의 좌상 위치는  $((2m-1)*64, (2n+1)*64)$ 로 표현되고,  $m$  및  $n$ 은 정수이며 현재 비디오 블록의 크기는  $64 \times 64$ 이다.
- [1240] I12. I1 항의 방법에 있어서, 현재 비디오 블록의 좌상 위치가  $((2m+1)*64, 2n*64)$ 로 표현되면, 재생 블록의 좌상 위치는  $(2m*64, 2n*64)$ 로 표현되고,  $m$  및  $n$ 은 정수이며 현재 비디오 블록의 크기는  $64 \times 64$ 이다.
- [1241] I13. I1 항의 방법에 있어서, 현재 비디오 블록의 좌상 위치가  $((2m+1)*64, 2n*64)$ 로 표현되면, 재생 블록의 좌상 위치는 현재 비디오 블록의 좌상 위치에 위치하고,  $m$  및  $n$ 은 정수이며 현재 비디오 블록의 크기는  $64 \times 64$ 이다.
- [1242] I14. I1 항의 방법에 있어서, 현재 비디오 블록의 좌상 위치가  $(2m*64, (2n+1)*64)$ 로 표현되면, 재생 블록의 좌상 위치는  $((2m-1)*64, (2n+1)*64)$ 로 표현되고,  $m$  및  $n$ 은 정수이며 현재 비디오 블록의 크기는  $64 \times 64$ 이다.
- [1243] I15. I1 항의 방법에 있어서, 현재 비디오 블록의 좌상 위치가  $(2m*64, (2n+1)*64)$ 로 표현되면, 재생 블록의 좌상 위치는  $(2m*64, 2n*64)$ 로 표현되고,  $m$  및  $n$ 은 정수이며 현재 비디오 블록의 크기는  $64 \times 64$ 이다.
- [1244] I16. I1 항의 방법에 있어서, 현재 비디오 블록의 좌상 위치가  $(2m*64, (2n+1)*64)$ 로 표현되면, 재생 블록의 좌상 위치는  $((2m+1)*64, 2n*64)$ 로 표현되고,  $m$  및  $n$ 은 정수이며 현재 비디오 블록의 크기는  $64 \times 64$ 이다.
- [1245] I17. I1 항의 방법에 있어서, 현재 비디오 블록의 좌상 위치가  $(2m*64, (2n+1)*64)$ 로 표현되면, 재생 블록의 좌상 위치는 현재 비디오 블록의 좌상 위치에 위치하고,  $m$  및  $n$ 은 정수이며 현재 비디오 블록의 크기는  $64 \times 64$ 이다.
- [1246] J1. 시각 매체 처리 방법에 있어서,
- [1247] 시각 매체 데이터의 현재 비디오 블록 및 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 인트라 블록 복사 모드에서 예측을 위한 참조 샘플들을 저장하는 버퍼를 결정하는 단계, 여기서 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생된 블록과 연관된 움직임 정보에 기반한 인트라 블록 복사 모드에서 수행하고,
- [1248] 현재 비디오 블록을 포함하는 코딩 유닛(CU)에 연관된 현재 비디오 블록의 위치  $(x_0, y_0)$ 에 공간적으로 위치한 샘플에 대해 버퍼에서 참조 위치  $(P, Q)$ 에서 시작하여 대응되는 참조 영역을 산출하는 단계, 및
- [1249] 이전에 처리된 블록 중 예측에 사용되는 블록을 결정하기 위해 참조 영역 및 참조 위치를 조정하는 단계를 포함한다.
- [1250] J2. J1 항의 방법에 있어서,  $(y_0 > 6) \& 1 == 0$ 이면, 최대 두 개의 이전에 처리된 블록들이 예측을 위해 사용되며, 두 개의 이전에 처리된 블록들의 좌상 위치들은  $((x > 6 \& 6) - 128, y > 6 \& 6)$  및  $((x > 6 \& 6) - 64, y > 6 \& 6)$ 으로 표현된다.
- [1251] J3. J1 항의 방법에 있어서,  $(y_0 > 6) \& 1 == 1$ 이면, 좌상 위치가  $((x > 6 \& 6) - 64, y > 6 \& 6)$ 인 한 개의 이전에 처

리된 블록들이 예측을 위해 사용된다.

- [1252] K1. 비디오 처리 방법에 있어서,
- [1253] 비디오의 현재 비디오 블록 및 현재 비디오 블록의 비트스트림 표현 간의 변환을 위해 비디오의 구성요소 X를 이용하여 비디오의 구성요소 c의 현재 비디오 블록에 대응되는 블록 벡터의 유효성을 결정하는 단계, 여기서, 구성요소 X는 비디오의 루마 구성요소와 상이하고,
- [1254] 현재 비디오 블록에 대해 블록 벡터가 유효하다는 결정 하에 블록 벡터를 이용하여 변환을 수행하는 단계를 포함하되, 변환은 현재 비디오 블록을 갖는 동일 비디오 영역에 위치한 재생된 블록과 연관된 움직임 정보에 기반하여 인트라 블록 복사(IBC) 모드에서 수행된다.
- [1255] K2. K1 항의 방법에 있어서, 현재 비디오 블록은 위치 (x, y)에서 시작하고, 결정하는 단계는  $\text{isRec}(((x+Bv_x)>>6<<6)+128-(((y+Bv_y)>>6)\&1)*64+(x\%64), ((y+Bv_y)>>6<<6)+(y\%64))$ 이 참이면 블록 벡터가 유효하지 않은 것으로 결정되는 것을 포함하고, 여기서  $\text{isRec}(x,y)$ 는 샘플(x,y)이 IBC 모드로 재생된 경우 참이고, 블록 벡터는 (BVx, BVy)로 표시된다.
- [1256] K3. K1 항의 방법에 있어서, 구성요소 c는 비디오의 루마 구성요소에 대응된다.
- [1257] K4. K1 항의 방법에 있어서, 현재 비디오 블록은 크로마 블록이고 비디오는 4:4:4 포맷이다.
- [1258] K5. K1 항의 방법에 있어서, 현재 비디오 블록은 루마 구성요소 및 크로마 구성요소를 포함한다.
- [1259] K6. K1 항의 방법에 있어서, 비디오는 4:2:0 포맷이고, 현재 비디오 블록은 위치 (x, y)에서 시작하는 크로마 블록이고, 결정하는 단계는  $\text{isRec}(c, ((x+Bv_x)>>5<<5)+64-(((y+Bv_y)>>5)\&1)*32+(x\%32), ((y+Bv_y)>>5<<5)+(y\%32))$ 이 참이면 블록 벡터가 유효하지 않은 것으로 결정하는 것을 포함하며,  $\text{isRec}(x,y)$ 는 샘플(x,y)이 IBC 모드로 재생된 경우 참이다.
- [1260] K7. K1 항의 방법에 있어서, 결정하는 단계는 비디오의 구성요소 X의 샘플들의 가용성에 적어도 부분적으로 기초한다.
- [1261] K8. K7 항의 방법에 있어서, 현재 비디오 블록은 위치 (x, y)에서 시작하고, 결정하는 단계는  $\text{isRec}(c, ((x+Bv_x)>>6<<6)+128-(((y+Bv_y)>>6)\&1)*64+(x\%64), ((y+Bv_y)>>6<<6)+(y\%64))$ 이 참이면 블록 벡터가 유효하지 않은 것으로 결정되는 것을 포함하고, 여기서  $\text{isRec}(c, x,y)$ 는 샘플(x,y)에 대한 구성요소 c가 가용하고, IBC 모드로 재생된 경우 참이고, 블록 벡터는 (BVx, BVy)로 표시된다.
- [1262] K9. K8 항의 방법에 있어서, 현재 비디오 블록은 루마 블록이다.
- [1263] K10. K8 항의 방법에 있어서, 현재 비디오 블록은 크로마 블록이고 비디오는 4:4:4 포맷이다.
- [1264] K11. K7 항의 방법에 있어서, 결정하는 단계는 비디오의 구성요소 X의 샘플들의 가용성을 결정하는 것을 포함한다.
- [1265] K12. K7 항의 방법에 있어서, 현재 비디오 블록은 위치 (x, y)에서 시작하고, 결정하는 단계는  $\text{isRec}(c, x+Bv_x+\text{Chroma\_CTU\_size}, y)$ 이면 블록 벡터가 유효하지 않은 것으로 결정되는 것을 포함하고, 여기서  $\text{isRec}(c, x,y)$ 는 샘플(x,y)에 대한 구성요소 c가 가용하고, IBC 모드로 재생된 경우 참이고, 블록 벡터는 (BVx, BVy)로 표시되며, Chroma\_CTU\_size는 크로마 구성요소에 대한 코딩 트리 유닛의 크기를 지시한다.
- [1266] K13. K12 항의 방법에 있어서, 크로마 구성요소에 대한 코딩 트리 유닛의 크기는 64이다.
- [1267] K14. K1 항의 방법에 있어서,
- [1268] 인트라 블록 복사 모드에서 예측을 위한 참조 샘플들을 저장하는 버퍼를 결정하는 단계; 및
- [1269] 현재 비디오 블록을 포함하는 코딩 트리 유닛(CTU)의 좌상단 위치에 대한 현재 비디오 블록의 위치 (x0, y0)에 공간적으로 위치한 샘플에 대해, 버퍼에서 참조 위치 (P, Q)에서 시작하는 대응 참조 영역을 산출하는 단계를 더 포함한다.
- [1270] K15. K14 항의 방법에 있어서, 버퍼는 크기가 MxM인 블록들을 저장한다.
- [1271] K16. K14 항의 방법에 있어서, 버퍼는 크기가 NxM인 블록들을 저장하되, M 및 N은 동일하지 않다.
- [1272] K17. K14 항의 방법에 있어서, 버퍼는 하나 이상의 조건이 충족되면 사용이 제한된다.

- [1273] K18. K14 항의 방법에 있어서, 버퍼가 현재 비디오 블록과 동일한 브릭/타일/타일 그룹/슬라이스 내에 있는 경우 버퍼는 사용이 제한된다.
- [1274] L1. A1-K18 항 중 어느 한 하위항에 있어서,
- [1275] 변환은 현재 비디오 블록으로부터 비트스트림 표현을 생성하는 것을 포함한다.
- [1276] L2. A1 항 내지 K18 항 중 어느 한 항의 방법에 있어서, 변환은 현재 비디오 블록의 픽셀 값들을 비트스트림 표현으로부터 생성하는 것을 포함한다.
- [1277] L3. 비디오 인코더 장치에 있어서, A1 항 내지 K18 항 중 하나 이상의 방법을 구현하는 프로세서를 포함한다.
- [1278] L4. 비디오 디코더 장치에 있어서, A1 항 내지 K18 항 중 하나 이상의 방법을 구현하는 프로세서를 포함한다.
- [1279] L5. 컴퓨터 독출 가능한 매체에 있어서, A1 항 내지 K18 항 중 하나 이상의 방법을 구현하는 프로세서 실행 가능한 명령어들을 구현하는 코드를 저장한다.
- [1280] 이 명세서에서 "비디오 처리(video processing)"라는 용어는 비디오 인코딩, 비디오 디코딩, 비디오 압축, 또는 비디오 압축해제를 지칭할 수 있다. 예를 들어, 비디오 압축 알고리즘들이 비디오의 화소 표현으로부터 대응 비트스트림 표현으로의 변환 동안 적용될 수 있으며 그 역도 마찬가지이다. 현재 비디오 블록의 비트스트림 표현은 예를 들어 선택스에 정의된 바와 같이 비트스트림의 다른 위치들에 공존(co-located)하거나 분산된 비트들에 대응할 수 있다. 예를 들어 매크로블록(macroblock)은 변환 및 코딩된 오류 잔차 값(error residual value)들로, 또한 헤더 또는 비트스트림 내의 다른 필드들의 비트를 사용하여 인코딩될 수 있다.
- [1281] 이상에서, 현재 개시되는 (본 발명) 기술의 특정한 실시예들이 예시의 목적으로 기재되었지만 본 발명의 범위를 벗어나지 않고도 다양한 변경들이 이뤄질 수 있음을 이해해야 할 것이다. 이에 따라, 현재 개시되는 (본 발명) 기술은 첨부된 청구항들에 의해서가 아니면 한정되지 않는다.
- [1282] 이 명세서에 기재된 주제의 구현과 기능적 작동들은, 이 명세서에 개시된 구조들과 그 구조적 등가물들, 또는 그 하나 이상의 조합들을 포함하여 여러 가지 시스템, 디지털 전자 회로, 또는 컴퓨터 소프트웨어, 펌웨어, 또는 하드웨어로 구현될 수 있다. 이 명세서에 기재된 주제의 구현과 기능적 작동들은 하나 이상의 컴퓨터 프로그램 제품들, 즉 데이터 처리 장치에서 실행되거나 그 작동을 제어하는 유형 및 비일시적인 컴퓨터 독출 가능한 매체 상에 인코딩된 컴퓨터 프로그램 명령의 하나 이상의 모듈들로 구현될 수 있다. 컴퓨터 독출 가능한 매체는 기계 독출 가능한 저장 장치, 기계 독출 가능한 저장 기판, 메모리 장치, 기계 독출 가능한 전파 신호(propagated signal)를 일으키는(effecting) 물질의 조성(composition of matter), 또는 그 하나 이상의 조합이 될 수 있다. "데이터 처리 유닛(data processing unit)" 또는 "데이터 처리 장치(data processing apparatus)"라는 용어는 예를 들어 프로그래밍 가능한 프로세서, 컴퓨터, 또는 복수의 프로세서들 또는 컴퓨터들을 포함하여 데이터를 처리하는 모든 장치, 기기, 또는 기계들을 포괄한다. 장치는 하드웨어에 추가하여 예를 들어 프로세서 펌웨어를 구성하는 코드, 프로토콜 스택(protocol stack), 데이터베이스 관리 시스템, 운영체제, 또는 그 하나 이상의 조합 등 문제의 컴퓨터 프로그램에 대한 실행 환경을 생성하는 코드를 포함할 수 있다.
- [1283] (프로그램, 소프트웨어, 소프트웨어 앱(application), 스크립트(script), 또는 코드로도 알려진) 컴퓨터 프로그램은 컴파일형 및 해석형 언어들을 포함하는 어느 형태의 프로그래밍 언어로 기입되어, 독립형(stand-alone) 프로그램 또는 모듈, 컴포넌트, 서브루틴, 또는 연산 환경에 사용되기에 적합한 다른 유닛을 포함하는 어느 형태로 배치될 수 있다. 컴퓨터 프로그램이 반드시 파일(file) 또는 파일 시스템에 대응해야 하는 것은 아니다. 프로그램은 (예를 들어 마크업 언어 문서(markup language document)에 저장된 하나 이상의 스크립트들 등의) 다른 프로그램들 또는 데이터를 보유하는 파일의 일부에, 문제의 프로그램 전용의 단일한 프로그램에, 또는 (예를 들어 하나 이상의 모듈들, 서브프로그램들, 또는 코드의 일부들 등) 복수의 분산 파일들(multiple coordinated file)에 저장될 수 있다. 컴퓨터 프로그램은 하나의 컴퓨터 또는 하나의 장소(site)에 위치하거나 또는 통신 네트워크에 의해 상호연결된 복수의 장소들에 걸쳐 분포되는 복수의 컴퓨터들 상에서 실행되도록 배치될 수 있다.
- [1284] 이 명세서에 기재된 프로세스들과 로직 흐름들은 하나 이상의 컴퓨터 프로그램들을 실행하여 입력 데이터에 대해 작동되어 출력을 생성함으로써 기능들을 수행하는 하나 이상의 프로그래밍 가능한 프로세서들로 수행될 수 있다. 프로세스들과 로직 흐름들은 또한 예를 들어 FPGA(field programmable gate array; 현장 프로그래밍 가능한 게이트 어레이) 또는 ASIC(application specific integrated circuit; 주문형 집적회로) 등의 특수목적 논리회로에 의해 수행되거나 장치 역시 이로써 구현될 수 있다.
- [1285] 컴퓨터 프로그램의 실행에 적합한 프로세서는 예를 들어 범용 및 특수목적 마이크로프로세서들과, 어떤 종류의

디지털 컴퓨터의 어느 하나 이상의 프로세서들을 포함한다. 일반적으로, 프로세서는 독출 전용 메모리 또는 랜덤 액세스 메모리 또는 양자로부터 명령들과 데이터를 수신한다. 컴퓨터의 필수 요소는 명령들을 수행하는 프로세서와 명령들과 데이터를 저장하는 하나 이상의 메모리 장치들이다. 일반적으로, 컴퓨터는 또한, 예를 들어 자기, 자기 광학 디스크, 또는 광학 디스크 등 데이터를 저장하는 하나 이상의 대용량 저장 장치들을 포함하거나, 이로부터 데이터를 수신하고 이로 데이터를 전송하도록 작동상 결합되거나, 양자 모두일 것이다. 그러나 컴퓨터는 이런 장치들을 가질 필요 없다. 컴퓨터 프로그램 명령들과 데이터를 저장하기에 적합한 컴퓨터 독출 가능한 매체는 예를 들어 EPROM, EEPROM, 또는 플래시 메모리 장치 등의 반도체 메모리 장치를 포함하여 모든 형태의 비휘발성 메모리, 매체, 및 메모리 장치들을 포함한다. 프로세서와 메모리는 특수목적 논리 회로로 보충되거나 이에 포함될 수 있다.

[1286] 도면들과 함께 이 명세서는 단지 예시로 간주될 것을 의도한 것인데, 여기서 예시(exemplary)는 예(example)를 의미한다. 이 명세서에 사용된 바와 같이, "또는(or)"의 사용은 문맥이 명확히 달리 지시하지 않는 한 "및/또는(and/or)"를 포함하려 의도한 것이다.

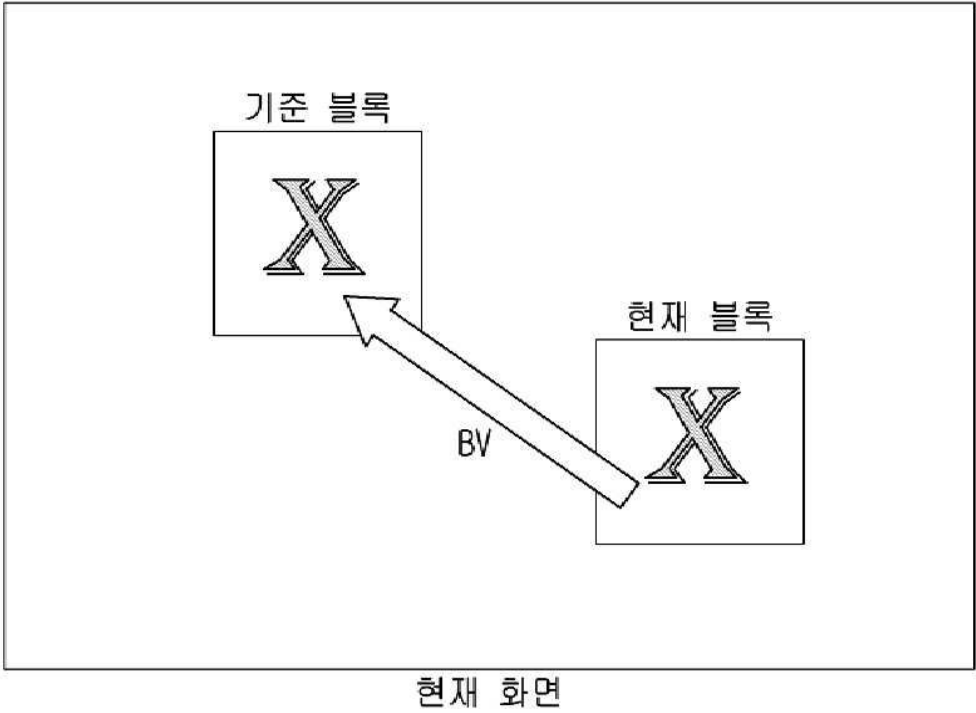
[1287] 이 명세서가 많은 상세들을 포함하지만, 이들은 어느 발명 또는 청구될 수 있는 것의 범위에 대한 제한으로 해석되어서는 안 되며, 특정한 발명의 특정한 실시예들에 특정될 수 있는 특징들의 설명으로 해석되어야 할 것이다. 이 명세서에 별도의 실시예들의 맥락에서 설명된 어떤 특징들은 단일한 실시예에 조합되어 구현될 수 있다. 역으로 단일한 실시예에 기재된 여러 특징들 역시 복수의 실시예들에 별도로 또는 어느 적절한 하부조합으로 구현될 수 있다. 뿐만 아니라, 특징들이 어떤 조합들로 동작하도록 위에 기재되고 심지어 처음 그렇게 청구되었더라도, 어떤 경우에는 청구된 조합으로부터 하나 이상의 특징들이 제거되어, 청구된 조합이 하부조합 또는 하부조합의 변형을 지향할 수 있다.

[1288] 마찬가지로, 작동들이 도면들에 특정한 순서로 도시되었지만, 바람직한 결과를 달성하기 위해 이러한 작동들이 도시된 특정한 순서 또는 순차적 순서로 실행되거나 모든 도시된 작동들이 실행될 것을 요구하는 것으로 이해해서는 안 된다. 뿐만 아니라, 이 명세서에 기재된 실시예들의 여러 가지 시스템 구성요소들의 분리가 모든 실시예들에서 이러한 분리를 요구하는 것으로 이해되어서는 안 된다.

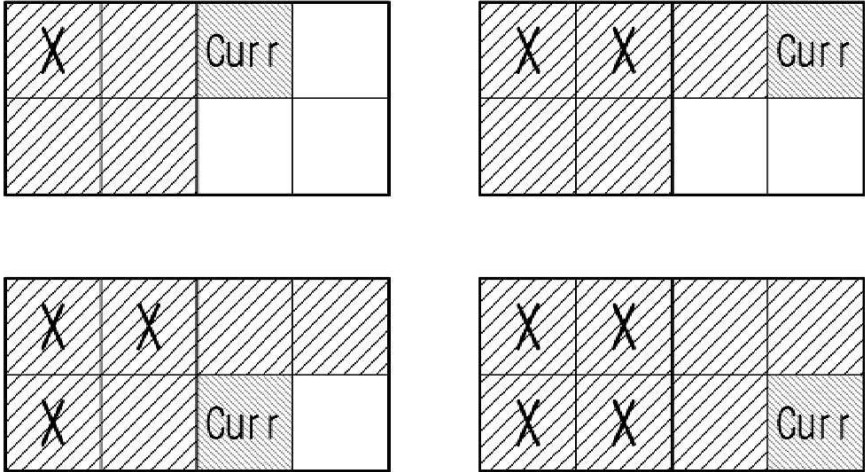
[1289] 일부만의 구현예와 예들이 도시되었지만, 다른 구현예들, 개선 및 변형들이 이 명세서에 설명되고 도시된 것에 기반하여 이뤄질 수 있을 것이다.

도면

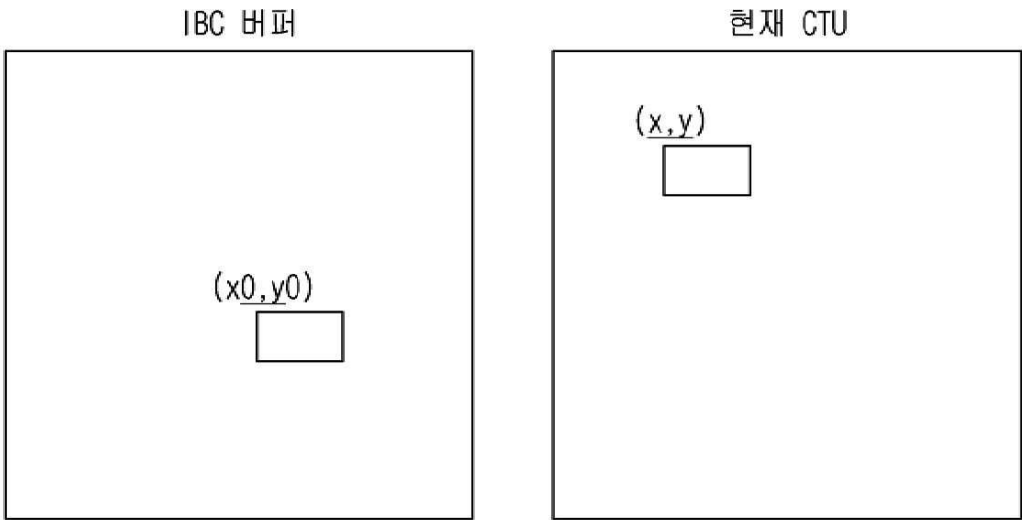
도면1



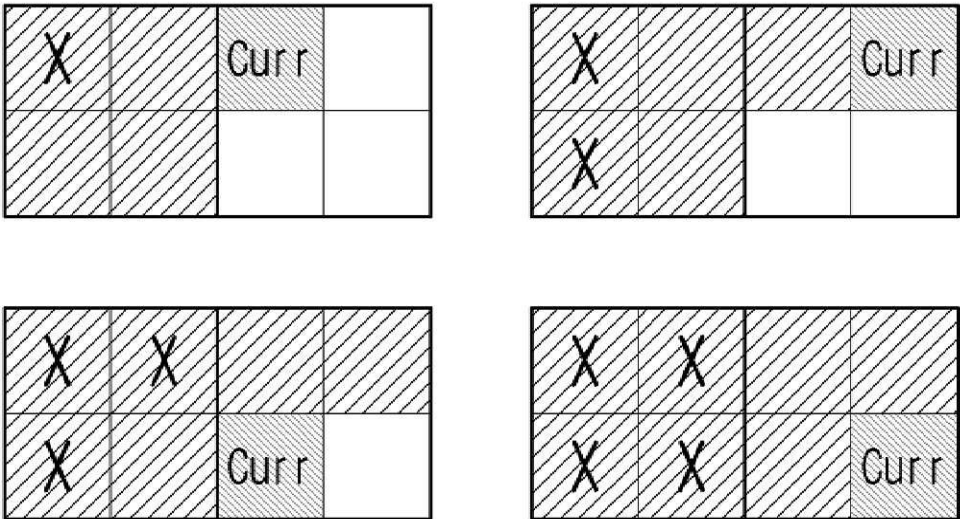
도면2



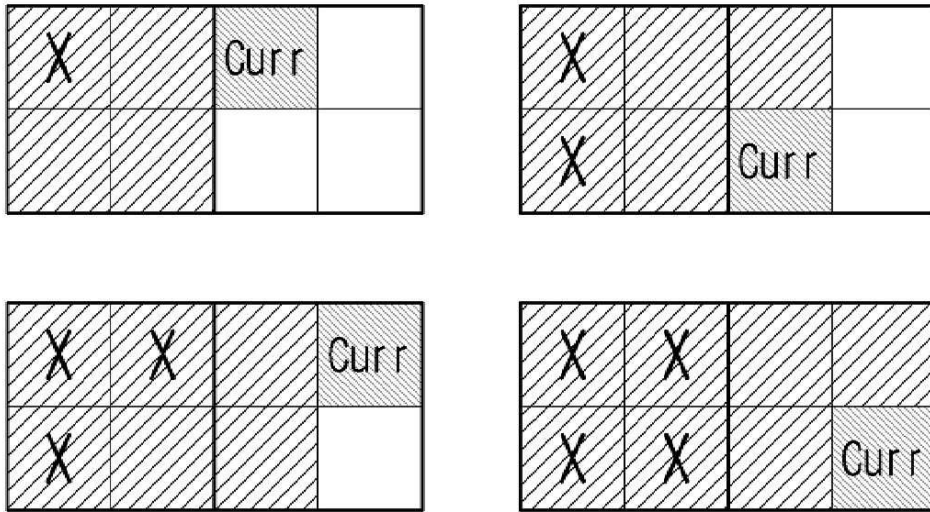
도면3



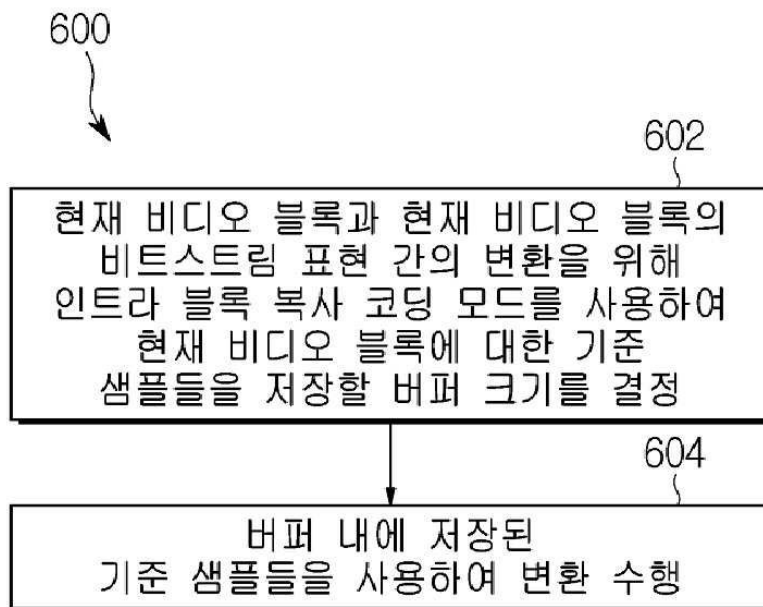
도면4



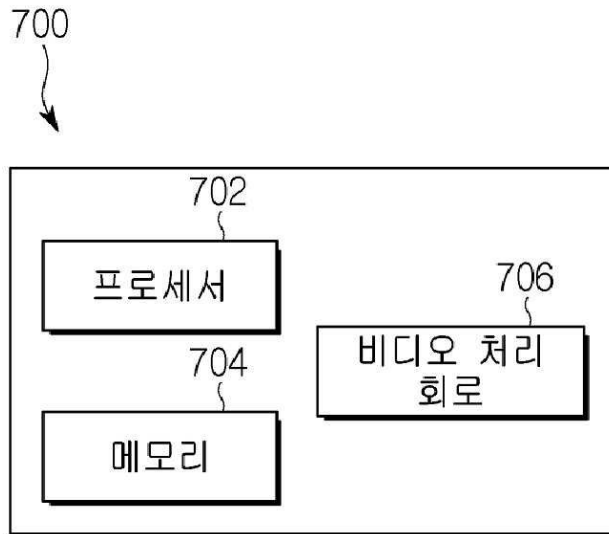
도면5



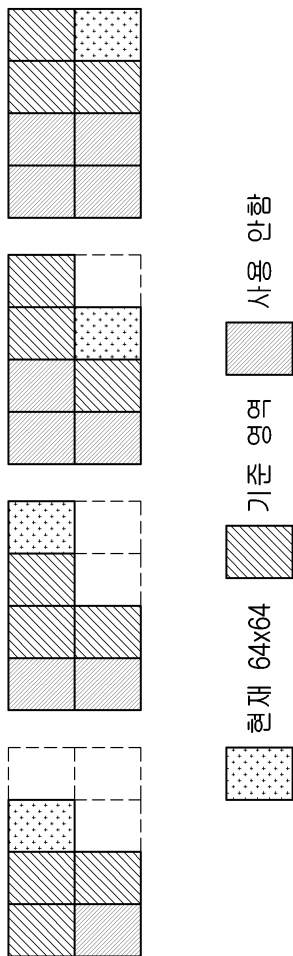
도면6



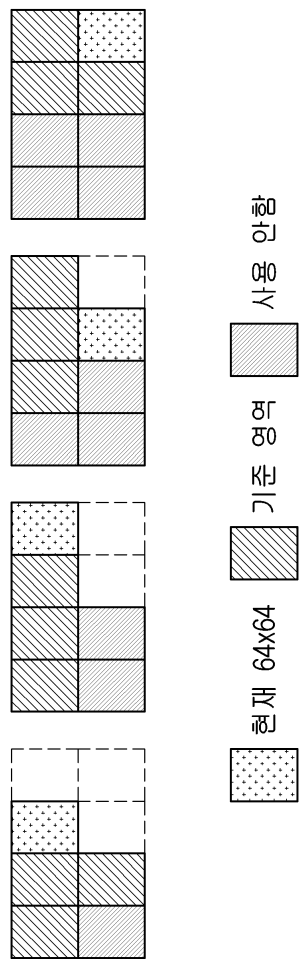
도면7



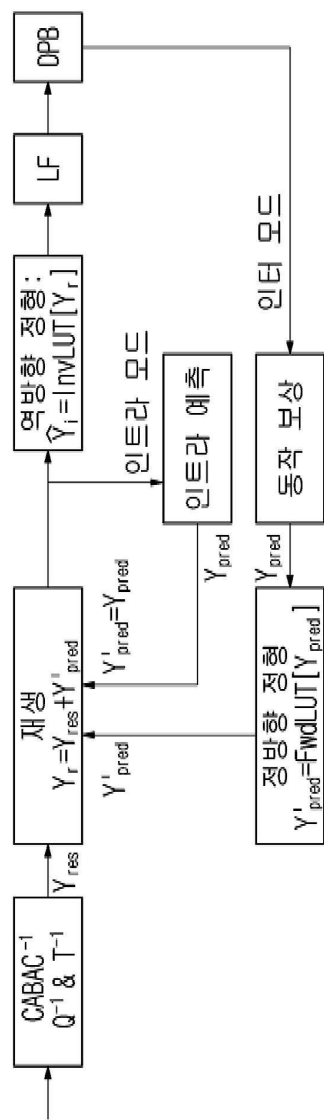
도면8



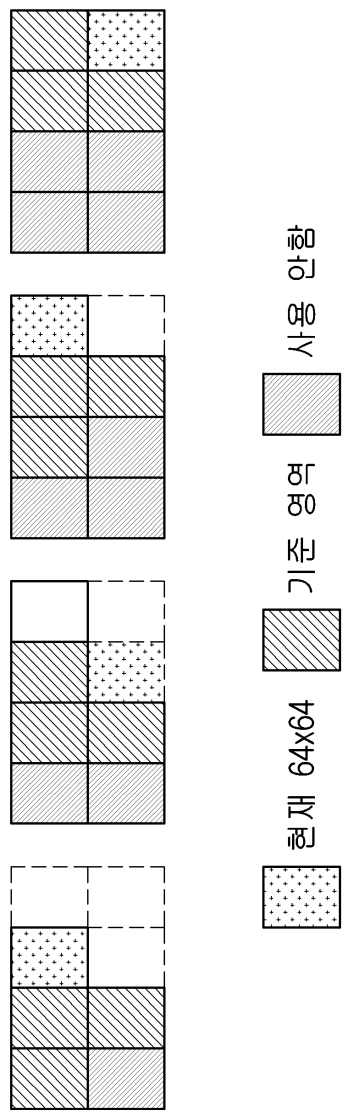
도면9



도면10



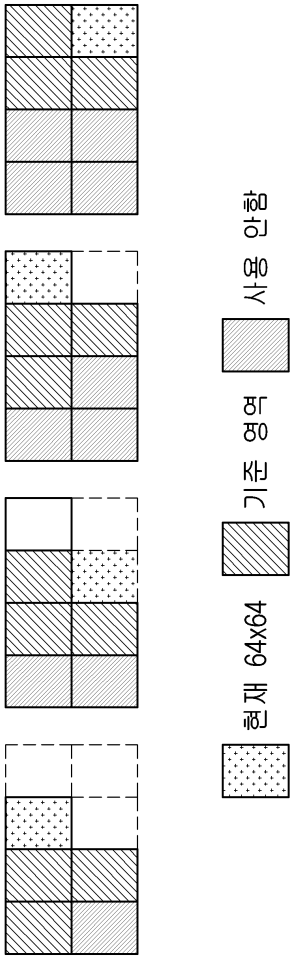
도면11



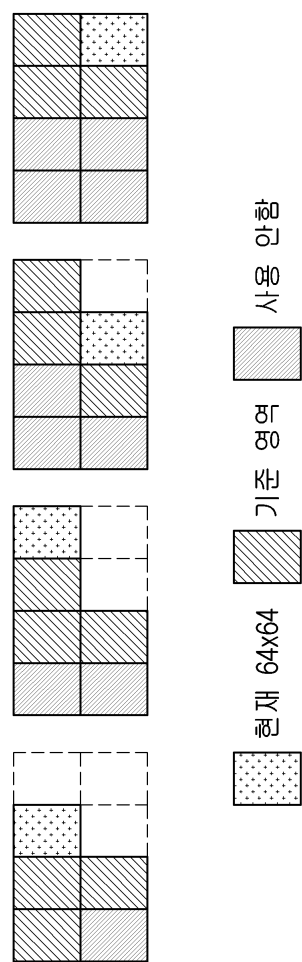
도면12



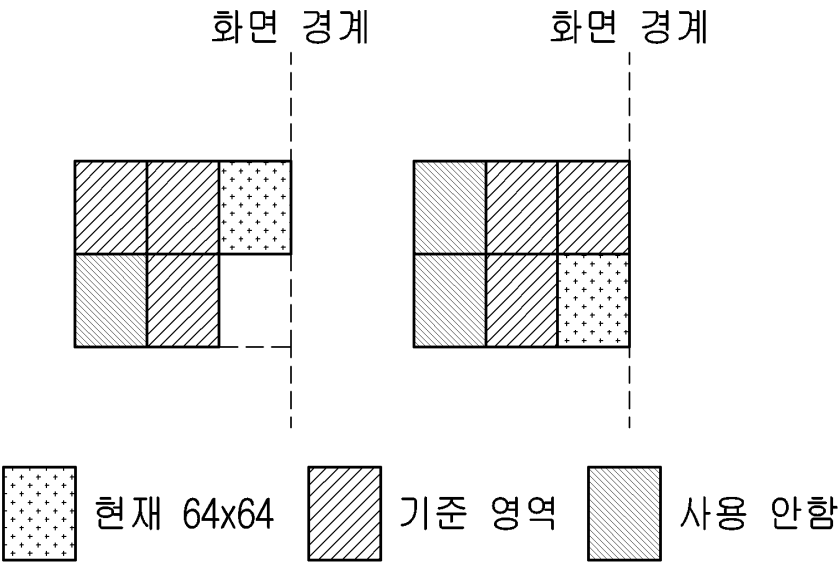
도면13



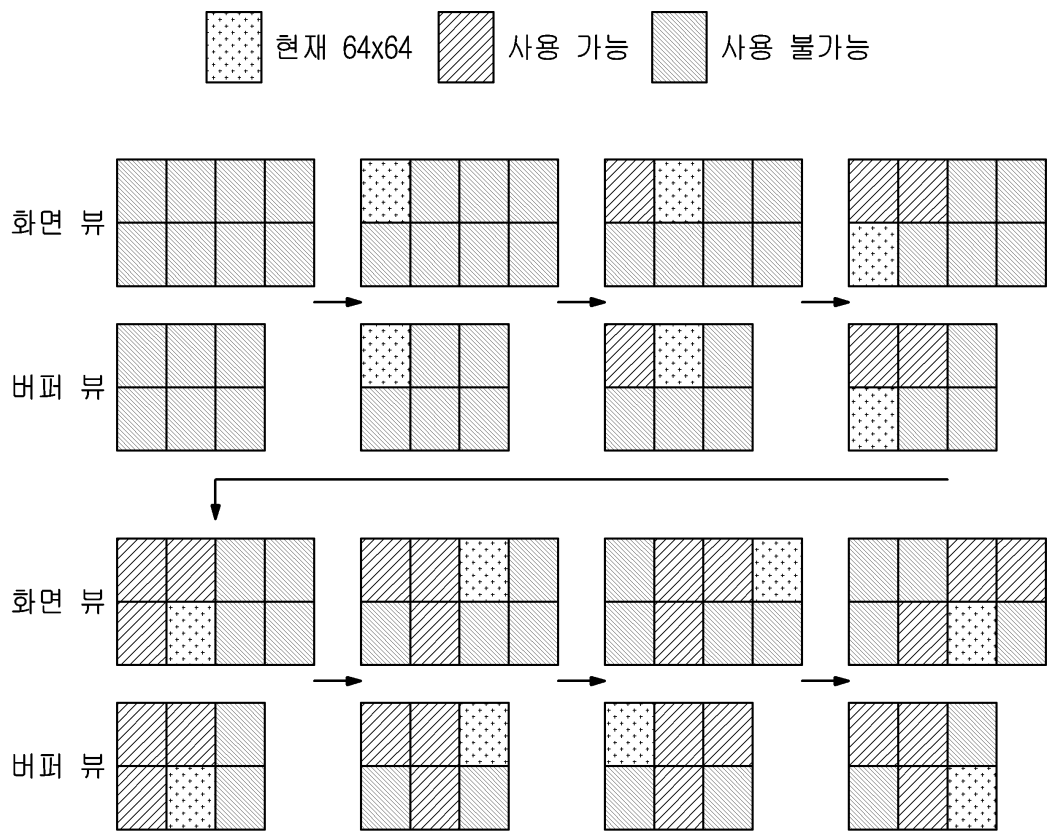
도면14



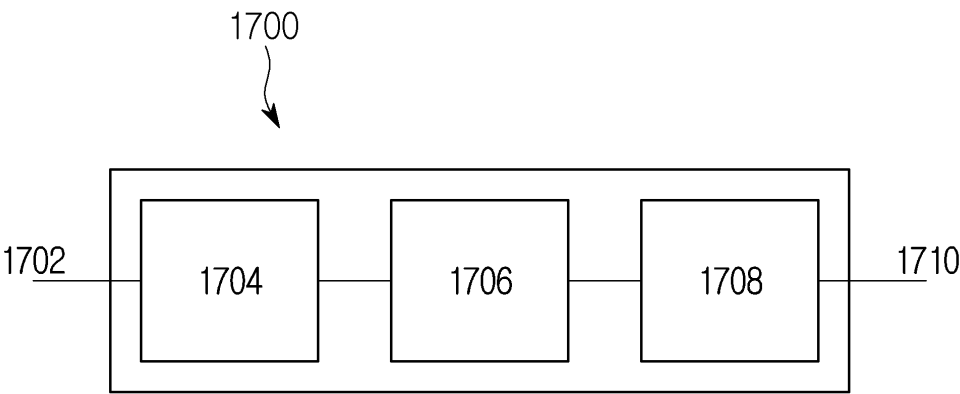
도면15



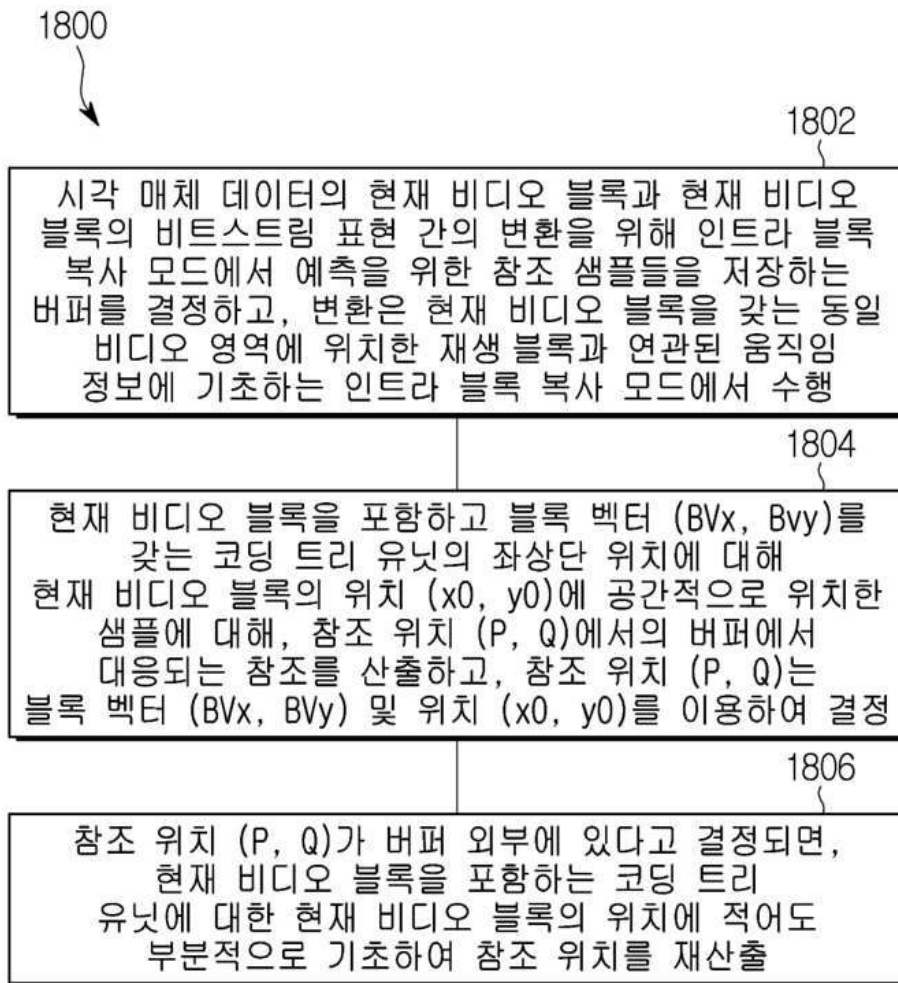
도면16



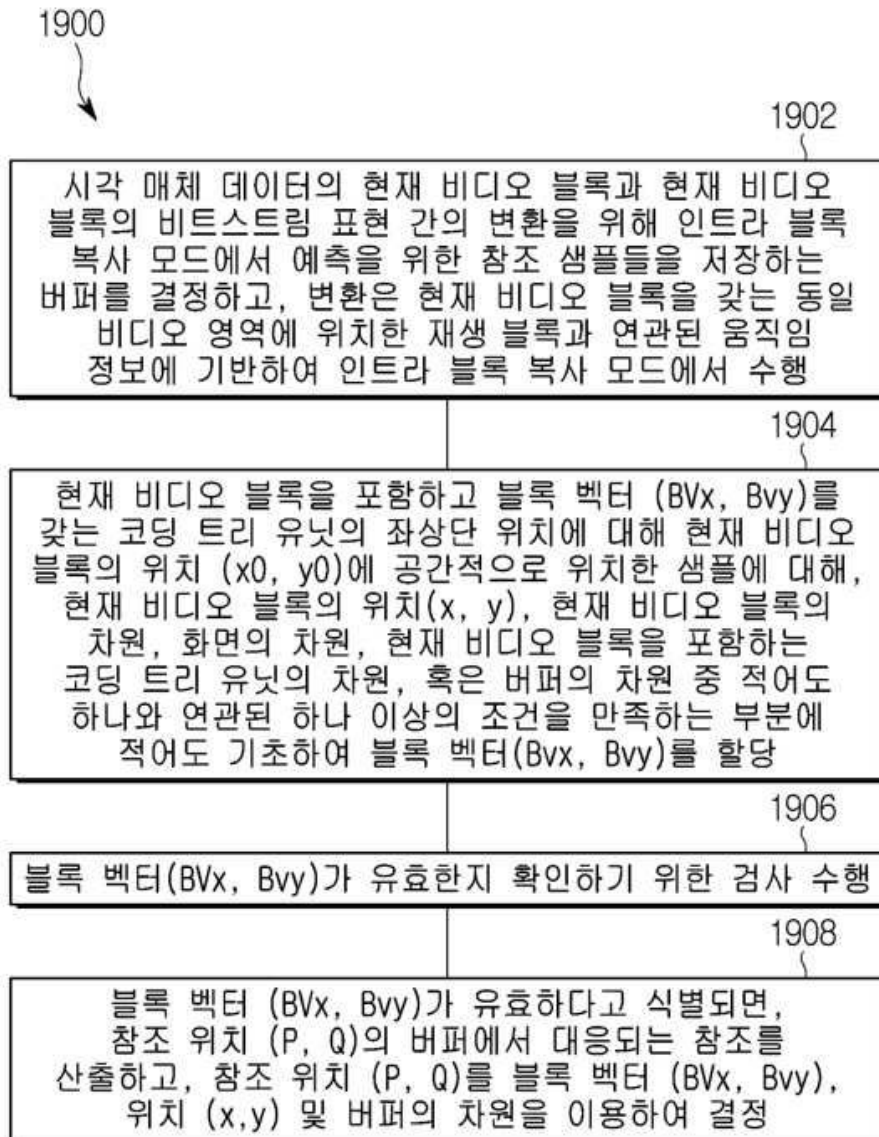
도면17



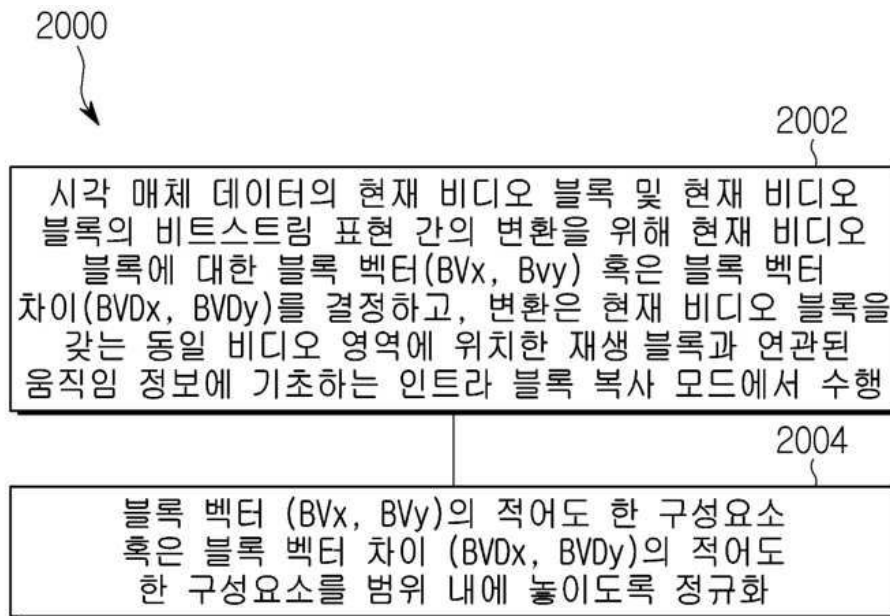
도면18



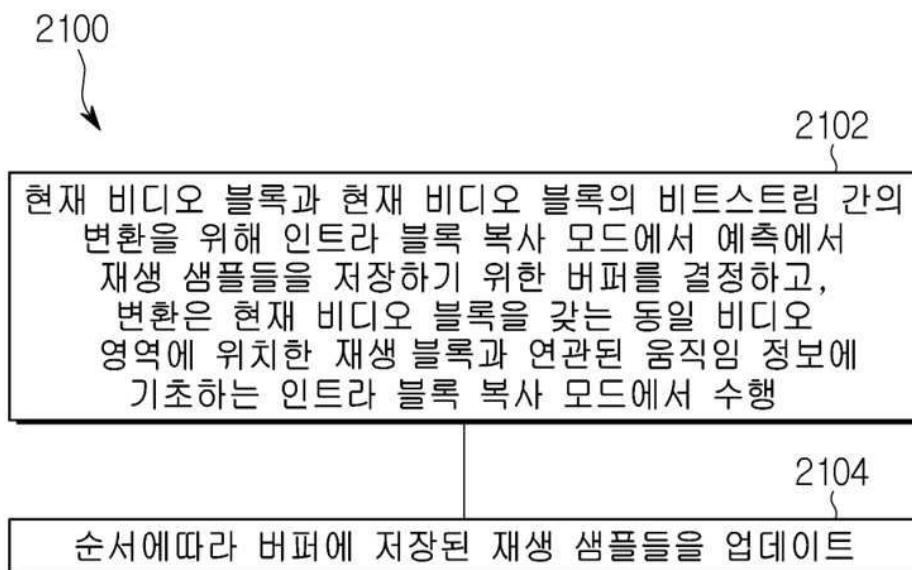
도면19



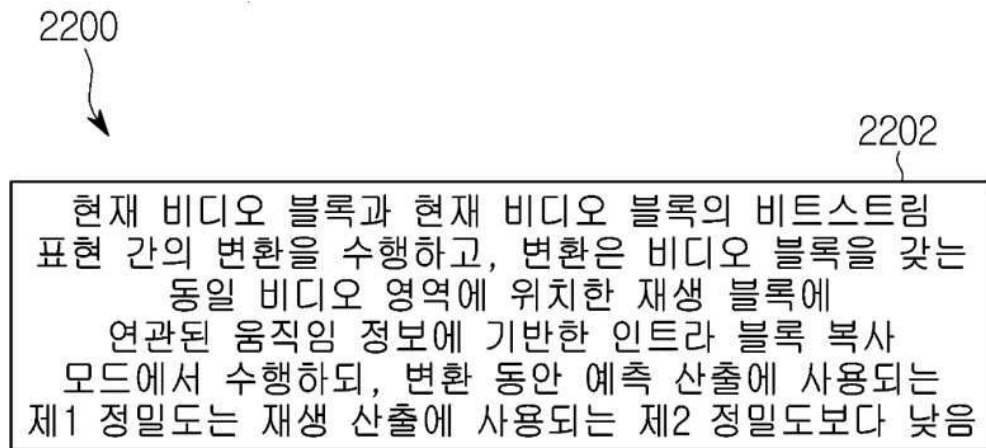
도면20



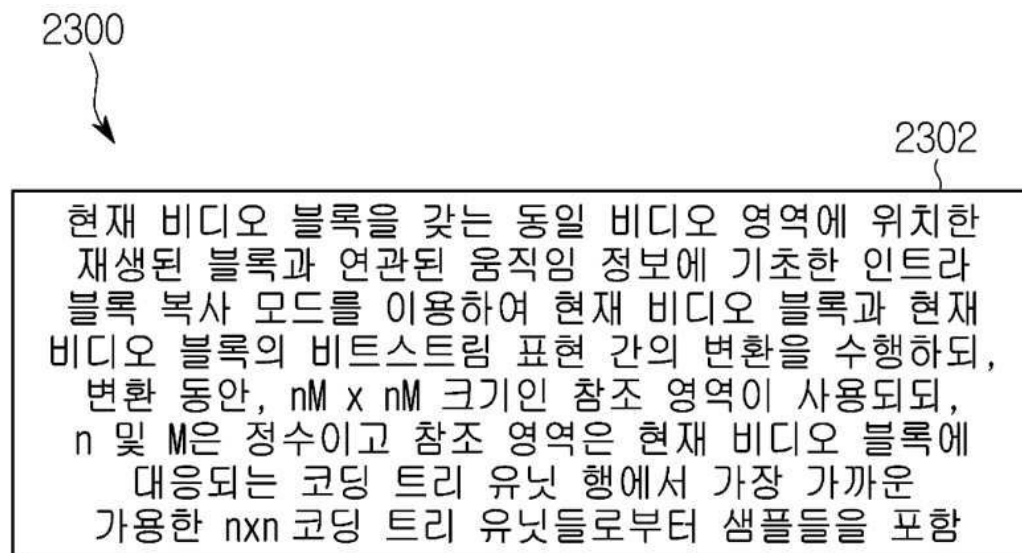
도면21



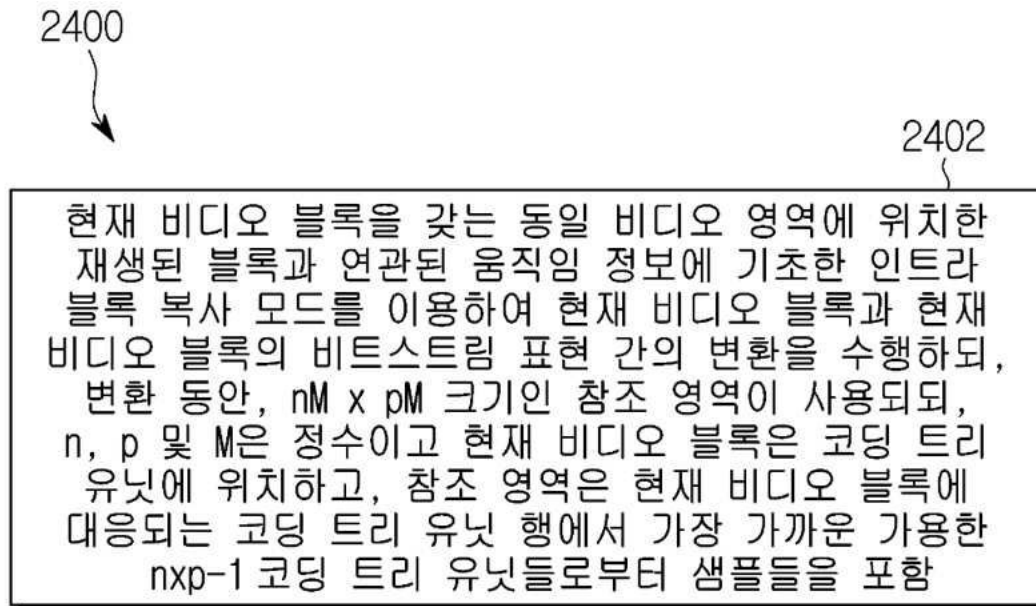
도면22



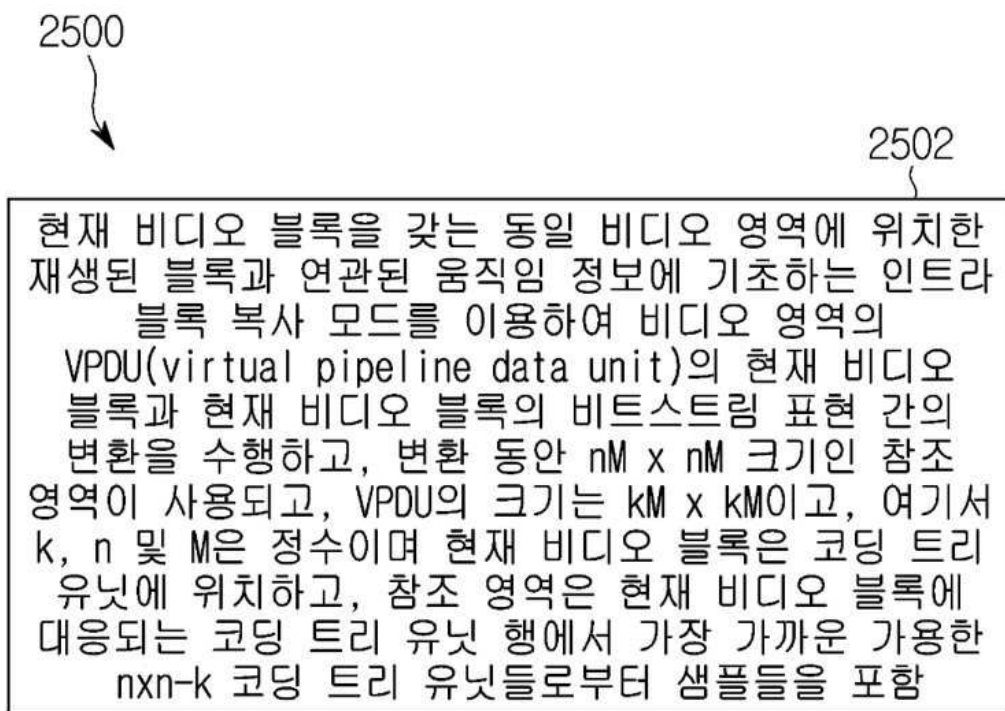
도면23



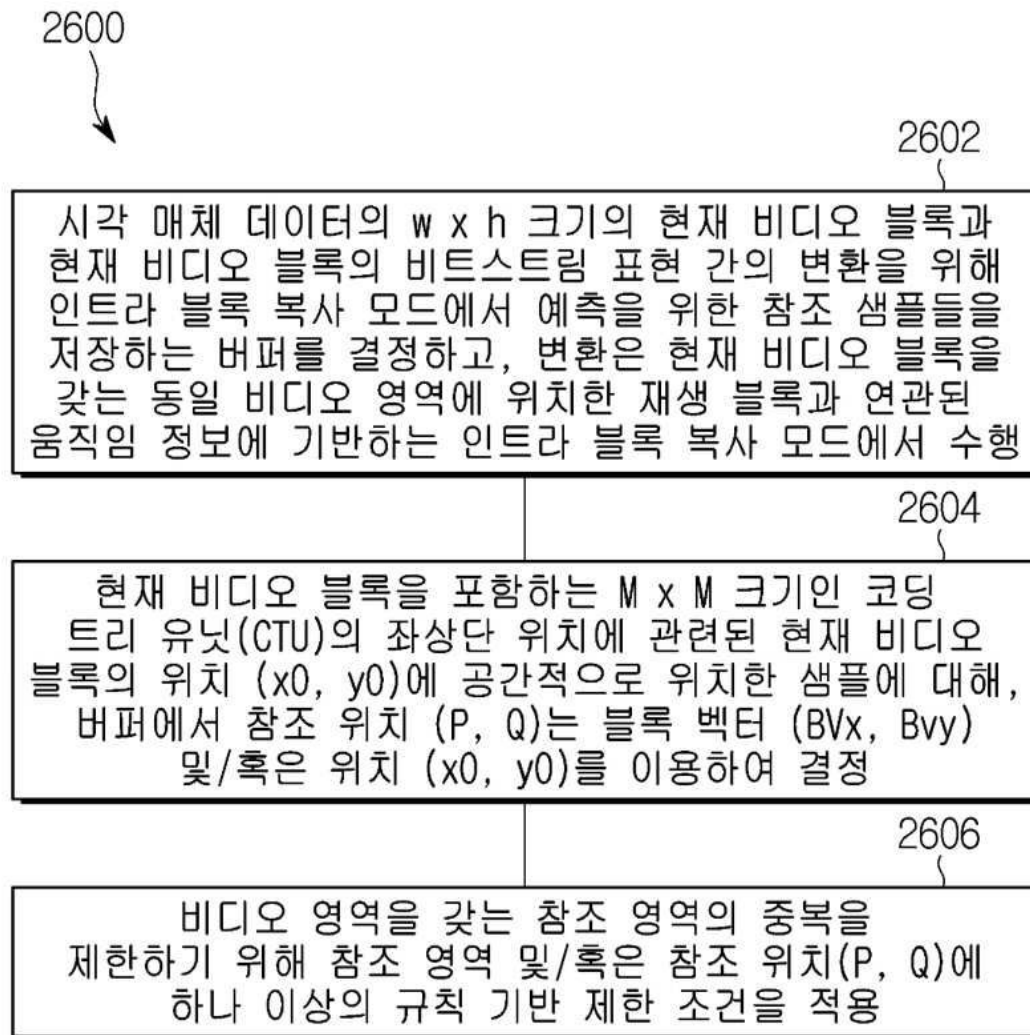
도면24



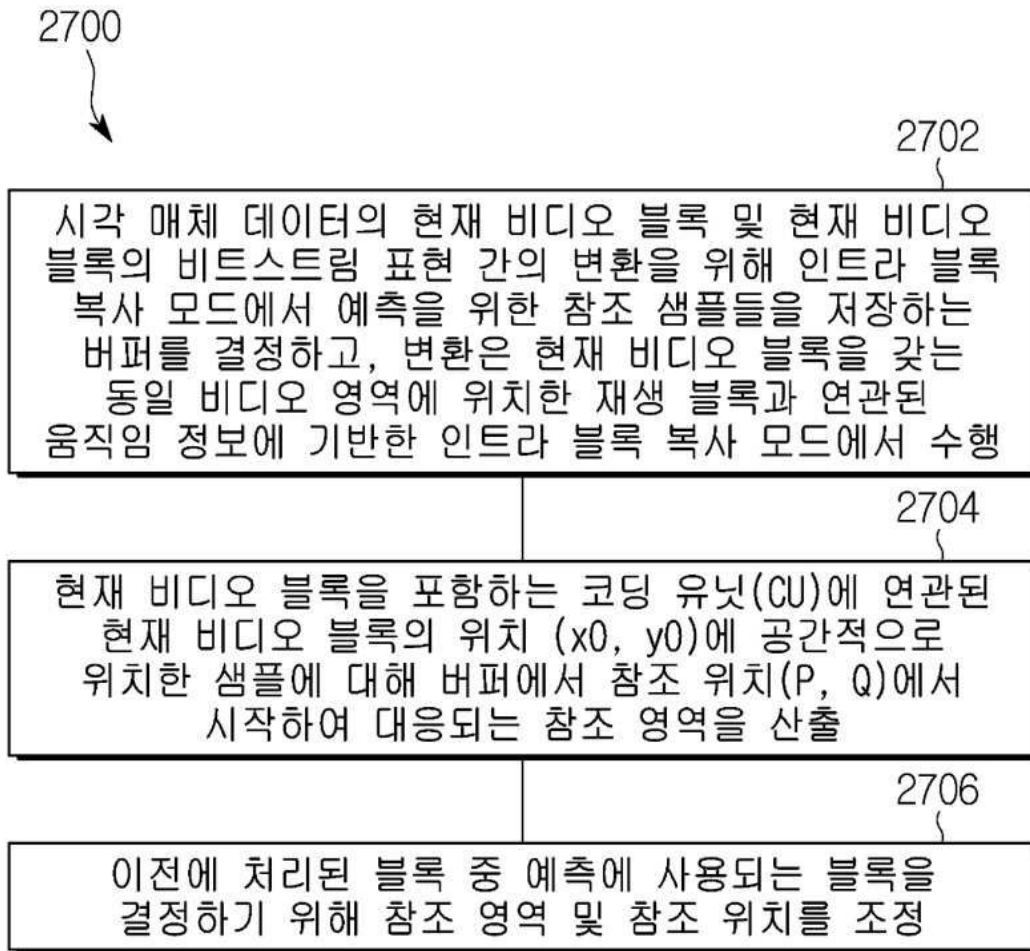
도면25



도면26



도면27



도면28

