(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2018-0098158
(43) 공개일자 2018년09월03일

(51) 국제특허분류(Int. Cl.)
H04N 19/96 (2014.01) *H04N 19/119* (2014.01)
H04N 19/172 (2014.01) *H04N 19/174* (2014.01)
H04N 19/70 (2014.01)
(52) CPC특허분류
H04N 19/96 (2015.01)
H04N 19/119 (2015.01)
(21) 출원번호 10-2018-0021636
(22) 출원일자 2018년02월23일
심사청구일자 없음
(30) 우선권주장
1020170024640 2017년02월24일 대한민국(KR)

(71) 출원인
주식회사 케이티
경기도 성남시 분당구 불정로 90(정자동)
(72) 발명자
이배근
서울특별시 서초구 태봉로 151 한국통신연구개발본부
(74) 대리인
최윤서

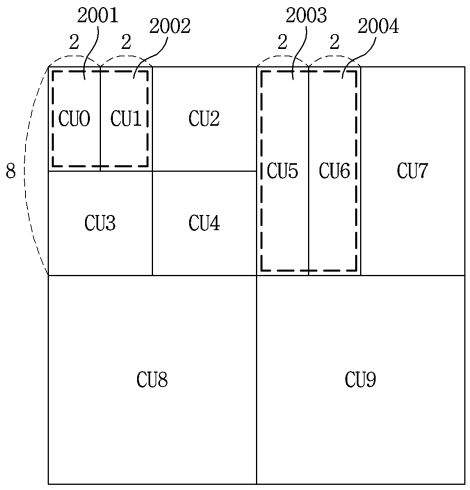
전체 청구항 수 : 총 22 항

(54) 발명의 명칭 **비디오 신호 처리 방법 및 장치**

(57) 요 약

본 발명은 비디오 신호 처리 방법 및 장치에 관한 것이다. 본 발명에 따른 영상 복호화 방법은, 현재 코딩 블록에 대한 블록 파티션 정보를 이용하여, 현재 코딩 블록이 이웃 병렬 처리 블록과 적어도 수평, 수직 또는 대각선 방향 중 어느 하나로 인접한 현재 병렬 처리 블록인지를 확인하는 단계, 및 상기 현재 병렬 처리 블록에 대해 상기 이웃 병렬 처리 블록과 함께 병렬 복호화하는 단계를 포함할 수 있다. 본 발명에 의하면, 부호화/복호화 대상 블록을 독립적으로 병렬 처리함으로써, 영상 신호의 부호화/복호화 효율을 증가시킬 수 있다.

대 표 도 - 도20



(52) CPC특허분류
H04N 19/172 (2015.01)
H04N 19/174 (2015.01)
H04N 19/70 (2015.01)

명 세 서

청구범위

청구항 1

현재 코딩 블록에 대한 블록 파티션 정보를 이용하여, 현재 코딩 블록이 이웃 병렬 처리 블록과 적어도 수평, 수직 또는 대각선 방향 중 어느 하나로 인접한 현재 병렬 처리 블록인지를 확인하는 단계, 및

상기 현재 병렬 처리 블록에 대해 상기 이웃 병렬 처리 블록과 함께 병렬 복호화하는 단계를 포함하는, 영상 복호화 방법.

청구항 2

제1 항에 있어서,

현재 병렬 처리 블록 주변으로 복원된 참조 샘플이 적어도 하나 이상 존재하지 않으면, 상기 이웃 병렬 처리 블록 주변의 복원된 참조 샘플을 활용하여 현재 병렬 처리 블록의 참조 샘플을 획득하는 단계를 더 포함하는, 영상 복호화 방법.

청구항 3

제2 항에 있어서,

상기 현재 병렬 처리 블록의 참조 샘플을 구하는 단계는,

상기 현재 병렬 처리 블록의 기준점 샘플 값과 상기 이웃 병렬 처리 블록의 기준점 샘플 간의 차이를 오프셋으로 구하는 단계, 및

상기 이웃 병렬 처리 블록 주변의 복원된 참조 샘플로부터 상기 오프셋을 차감하여 현재 병렬 처리 블록의 참조 샘플을 구하는 단계를 포함하는, 영상 복호화 방법.

청구항 4

제3 항에 있어서,

상기 현재 병렬 처리 블록의 너비(w) 크기가 높이(h) 크기 보다 작은 경우에는, 상기 현재 병렬 처리 블록의 기준점 샘플 위치는 복원된 상단 샘플을 활용하는, 영상 복호화 방법.

청구항 5

제3 항에 있어서,

상기 현재 병렬 처리 블록이 높이(h) 크기가 너비(w) 크기 보다 작은 경우에는, 상기 현재 병렬 처리 블록의 기준점 샘플 위치는 복원된 좌측 샘플을 활용하는 영상 복호화 방법.

청구항 6

제1 항에 있어서,

현재 병렬 처리 블록의 참조 샘플은, 이웃 병렬 처리 블록의 복원된 참조 샘플을 그대로 활용하는, 영상 복호화 방법.

청구항 7

제1 항에 있어서,

상기 현재 코딩 블록이, 병렬 코딩 유닛 단위 PCUR 영역내에 위치하면, 현재 병렬 처리 블록으로 결정하는, 영상 복호화 방법.

청구항 8

제7 항에 있어서,

상기 병렬 코딩 유닛 단위 PCUR의 형태는 부호화 과정에서 정방형 또는 비정방형 중 선택 가능하고, 상기 선택된 PCUR의 형태를 나타내는 신택스 요소를 획득하여, PCUR의 형태를 확인하는, 영상 복호화 방법.

청구항 9

제8 항에 있어서,

상기 병렬 코딩 유닛 단위 PCUR의 형태가 정방형으로 이루어 진 경우, 상기 선택된 정방형 PCUR의 크기를 나타내는 신택스 요소를 획득하여, 정방형 PCUR의 크기를 확인하는, 영상 복호화 방법.

청구항 10

제1 항에 있어서,

상기 병렬 복호화 단계는, 병렬 처리 블록 간에 적어도 모션 정보 및 참조 픽쳐 리스트의 공유 없이 독립적으로 복호화를 수행하는, 영상 복호화 방법.

청구항 11

제1 항에 있어서,

상기 병렬 복호화 단계는, 병렬 처리 블록 간에 적어도 인트라 참조 샘플의 공유 없이 독립적으로 복호화를 수행하는, 영상 복호화 방법.

청구항 12

제1 항에 있어서,

상기 현재 병렬 처리 블록은, 멀티 트리 파티셔닝에 의해 분할되어진 블록을 포함하는, 영상 복호화 방법.

청구항 13

현재 코딩 블록에 대한 블록 파티션 정보를 이용하여, 현재 코딩 블록이 이웃 병렬 처리 블록과 적어도 수평, 수직 또는 대각선 방향 중 어느 하나로 인접한 현재 병렬 처리 블록인지를 결정하는 단계, 및

상기 현재 병렬 처리 블록에 대해 상기 이웃 병렬 처리 블록과 함께 병렬 부호화하는 단계를 포함하는, 영상 부호화 방법.

청구항 14

제13 항에 있어서,

현재 병렬 처리 블록 주변으로 복원된 참조 샘플이 적어도 하나 이상 존재하지 않으면, 상기 이웃 병렬 처리 블록 주변의 복원된 참조 샘플을 활용하여 현재 병렬 처리 블록의 참조 샘플을 획득하는 단계를 더 포함하는, 영상 부호화 방법.

청구항 15

제14 항에 있어서,

상기 현재 병렬 처리 블록의 참조 샘플을 구하는 단계는,

상기 현재 병렬 처리 블록의 기준점 샘플 값과 상기 이웃 병렬 처리 블록의 기준점 샘플 간의 차이를 오프셋으로 구하는 단계, 및

상기 이웃 병렬 처리 블록 주변의 복원된 참조 샘플로부터 상기 오프셋을 차감하여 상기 현재 병렬 처리 블록의 참조 샘플을 구하는 단계를 포함하는, 영상 부호화 방법.

청구항 16

제14 항에 있어서,

상기 현재 병렬 처리 블록의 너비(w) 크기가 높이(h) 크기 보다 작은 경우에는, 상기 현재 병렬 처리 블록의 기준점 샘플 위치는 복원된 상단 샘플을 활용하는, 영상 부호화 방법.

청구항 17

제14 항에 있어서,

상기 현재 병렬 처리 블록이 높이(h) 크기가 너비(w) 크기 보다 작은 경우에는, 상기 현재 병렬 처리 블록의 기준점 샘플 위치는 복원된 좌측 샘플을 활용하는, 영상 부호화 방법.

청구항 18

제13 항에 있어서,

상기 현재 코딩 블록이, 기 결정된 병렬 코딩 유닛 단위 PCUR 영역내 위치하면, 현재 병렬 처리 블록으로 결정하는, 영상 부호화 방법.

청구항 19

제18 항에 있어서,

상기 병렬 코딩 유닛 단위 PCUR의 형태는 정방형 또는 비정방형 중 선택 가능하고, 상기 선택된 PCUR의 형태를 나타내는 신택스 요소를 시그날링하는, 영상 부호화 방법.

청구항 20

제19 항에 있어서,

상기 병렬 코딩 유닛 단위 PCUR의 형태가 정방형으로 선택된 경우, 상기 선택된 정방형 PCUR의 크기를 나타내는

신택스 요소를 시그날링하는, 영상 부호화 방법.

청구항 21

현재 코딩 블록에 대한 블록 파티션 정보를 이용하여, 현재 코딩 블록이 이웃 병렬 처리 블록과 적어도 수평, 수직 또는 대각선 방향 중 어느 하나로 인접한 현재 병렬 처리 블록인지를 확인하고, 상기 현재 병렬 처리 블록에 대해 상기 이웃 병렬 처리 블록과 함께 병렬 복호화하는 복호화기를 포함하는, 영상 복호화 장치.

청구항 22

영상 신호 비트스트림을 포함하는 기록매체에 있어서, 상기 기록매체에 포함된 영상 신호 비트스트림은, 현재 코딩 블록에 대한 블록 파티션 정보를 이용하여, 현재 코딩 블록이 이웃 병렬 처리 블록과 적어도 수평, 수직 또는 대각선 방향 중 어느 하나로 인접한 현재 병렬 처리 블록인지를 결정하는 단계, 및 상기 현재 병렬 처리 블록에 대해 상기 이웃 병렬 처리 블록과 함께 병렬 부호화하는 단계를 포함하는 영상 부호화 방법에 의해 부호화된 것을 특징으로 하는 기록매체.

발명의 설명

기 술 분 야

[0001] 본 발명은 비디오 신호 처리 방법 및 장치에 관한 것이다.

배 경 기 술

[0002] 최근 HD(High Definition) 영상 및 UHD(Ultra High Definition) 영상과 같은 고해상도, 고품질의 영상에 대한 수요가 다양한 응용 분야에서 증가하고 있다. 영상 데이터가 고해상도, 고품질이 될수록 기존의 영상 데이터에 비해 상대적으로 데이터량이 증가하기 때문에 기존의 유무선 광대역 회선과 같은 매체를 이용하여 영상 데이터를 전송하거나 기존의 저장 매체를 이용해 저장하는 경우, 전송 비용과 저장 비용이 증가하게 된다. 영상 데이터가 고해상도, 고품질화 됨에 따라 발생하는 이러한 문제들을 해결하기 위해서는 고효율의 영상 압축 기술들이 활용될 수 있다.

[0003] 영상 압축 기술로 현재 픽처의 이전 또는 이후 픽처로부터 현재 픽처에 포함된 화소값을 예측하는 화면 간 예측 기술, 현재 픽처 내의 화소 정보를 이용하여 현재 픽처에 포함된 화소값을 예측하는 화면 내 예측 기술, 출현 빈도가 높은 값에 짧은 부호를 할당하고 출현 빈도가 낮은 값에 긴 부호를 할당하는 엔트로피 부호화 기술 등 다양한 기술이 존재하고 이러한 영상 압축 기술을 이용해 영상 데이터를 효과적으로 압축하여 전송 또는 저장할 수 있다.

[0004] 한편, 고해상도 영상에 대한 수요가 증가함과 함께, 새로운 영상 서비스로서 입체 영상 컨텐츠에 대한 수요도 함께 증가하고 있다. 고해상도 및 초고해상도의 입체 영상 콘텐츠를 효과적으로 제공하기 위한 비디오 압축 기술에 대하여 논의가 진행되고 있다.

발명의 내용

해결하려는 과제

[0005] 본 발명은 영상 신호를 부호화/복호화함에 있어서, 부호화/복호화 대상 블록을 효과적으로 분할할 수 있는 멀티 트리 파티셔닝 방법 및 장치를 제공하는 것을 목적으로 한다.

[0006] 본 발명은 영상 신호를 부호화/복호화함에 있어서, 부호화/복호화 대상 블록을 대칭 형태 또는 비대칭 형태의 블록으로 분할하는 멀티 트리 파티셔닝 방법 및 장치를 제공하는 것을 목적으로 한다.

[0007] 본 발명은 멀티 트리 파티셔닝에 의해 분할된 코딩 블록을 병렬 부호화 및/또는 복호화 하는 방법 및 장치를 제공하는 것을 목적으로 한다.

[0008] 본 발명은 상기 부호화 방법에 의해 부호화된 영상 신호 비트스트림을 포함하는 기록매체를 제공하는 것을 목적으로 한다.

[0009] 본 발명에서 이루고자 하는 기술적 과제들은 이상에서 언급한 기술적 과제들로 제한되지 않으며, 언급하지 않은 또 다른 기술적 과제들은 아래의 기재로부터 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0010] 본 발명에 따른 영상 복호화 방법은, 현재 코딩 블록에 대한 블록 파티션 정보를 이용하여, 현재 코딩 블록이 이웃 병렬 처리 블록과 적어도 수평, 수직 또는 대각선 방향 중 어느 하나로 인접한 현재 병렬 처리 블록인지를 확인하는 단계, 및 상기 현재 병렬 처리 블록에 대해 상기 이웃 병렬 처리 블록과 함께 병렬 복호화하는 단계를 포함한다.

[0011] 또한, 상기 현재 병렬 처리 블록 주변으로 복원된 참조 샘플이 적어도 하나 이상 존재하지 않으면, 상기 이웃 병렬 처리 블록 주변의 복원된 참조 샘플을 활용하여 현재 병렬 처리 블록의 참조 샘플을 획득하는 단계를 더 포함한다.

[0012] 또한, 상기 현재 병렬 처리 블록의 참조 샘플을 구하는 단계는, 상기 현재 병렬 처리 블록의 기준점 샘플 값과 상기 이웃 병렬 처리 블록의 기준점 샘플 간의 차이를 오프셋으로 구하는 단계, 및 상기 이웃 병렬 처리 블록 주변의 복원된 참조 샘플로부터 상기 오프셋을 차감하여 현재 병렬 처리 블록의 참조 샘플을 구하는 단계를 포함한다.

[0013] 또한, 상기 현재 병렬 처리 블록의 너비(w) 크기가 높이(h) 크기 보다 작은 경우에는, 상기 현재 병렬 처리 블록의 기준점 샘플 위치는 복원된 상단 샘플을 활용한다.

[0014] 또한, 상기 현재 병렬 처리 블록이 높이(h) 크기가 너비(w) 크기 보다 작은 경우에는, 상기 현재 병렬 처리 블록의 기준점 샘플 위치는 복원된 좌측 샘플을 활용한다.

[0015] 또한, 현재 병렬 처리 블록의 참조 샘플은, 이웃 병렬 처리 블록의 복원된 참조 샘플을 그대로 활용한다.

[0016] 또한, 상기 현재 코딩 블록이, 병렬 코딩 유닛 단위 PCUR 영역내에 위치하면, 현재 병렬 처리 블록으로 결정한다.

[0017] 또한, 상기 병렬 코딩 유닛 단위 PCUR의 형태는 부호화 과정에서 정방형 또는 비정방형 중 선택 가능하고, 상기 선택된 PCUR의 형태를 나타내는 신택스 요소를 획득하여, PCUR의 형태를 확인한다.

[0018] 또한, 상기 병렬 코딩 유닛 단위 PCUR의 형태가 정방형으로 이루어 진 경우, 상기 선택된 정방형 PCUR의 크기를 나타내는 신택스 요소를 획득하여, 정방형 PCUR의 크기를 확인한다.

[0019] 또한, 상기 병렬 복호화 단계는, 병렬 처리 블록 간에 적어도 모션 정보 및 참조 픽처 리스트의 공유 없이 독립적으로 복호화를 수행한다.

[0020] 또한, 상기 병렬 복호화 단계는, 병렬 처리 블록 간에 적어도 인트라 참조 샘플의 공유 없이 독립적으로 복호화를 수행한다.

[0021] 또한, 상기 현재 병렬 처리 블록은, 멀티 트리 파티셔닝에 의해 분할되어진 블록을 포함한다.

[0022] 본 발명에 의한 영상 부호화 방법은, 현재 코딩 블록에 대한 블록 파티션 정보를 이용하여, 현재 코딩 블록이 이웃 병렬 처리 블록과 적어도 수평, 수직 또는 대각선 방향 중 어느 하나로 인접한 현재 병렬 처리 블록인지를 결정하는 단계, 및 상기 현재 병렬 처리 블록에 대해 상기 이웃 병렬 처리 블록과 함께 병렬 부호화하는 단계를 포함한다.

[0023] 또한, 현재 병렬 처리 블록 주변으로 복원된 참조 샘플이 적어도 하나 이상 존재하지 않으면, 상기 이웃 병렬 처리 블록 주변의 복원된 참조 샘플을 활용하여 현재 병렬 처리 블록의 참조 샘플을 획득한다.

[0024] 또한, 상기 현재 병렬 처리 블록의 참조 샘플을 구하는 단계는, 상기 현재 병렬 처리 블록의 기준점 샘플 값과 상기 이웃 병렬 처리 블록의 기준점 샘플 간의 차이를 오프셋으로 구하는 단계, 및 상기 이웃 병렬 처리 블록 주변의 복원된 참조 샘플로부터 상기 오프셋을 차감하여 상기 현재 병렬 처리 블록의 참조 샘플을 구하는 단계를 포함한다.

[0025] 또한, 상기 현재 병렬 처리 블록의 너비(w) 크기가 높이(h) 크기 보다 작은 경우에는, 상기 현재 병렬 처리 블록의 기준점 샘플 위치는 복원된 상단 샘플을 활용한다.

[0026] 또한, 상기 현재 병렬 처리 블록이 높이(h) 크기가 너비(w) 크기 보다 작은 경우에는, 상기 현재 병렬 처리 블록의 기준점 샘플 위치는 복원된 좌측 샘플을 활용한다.

[0027] 또한, 상기 현재 코딩 블록이, 기 결정된 병렬 코딩 유닛 단위 PCUR 영역내 위치하면, 현재 병렬 처리 블록으로 결정한다.

[0028] 또한, 상기 병렬 코딩 유닛 단위 PCUR의 형태는 정방형 또는 비정방형 중 선택 가능하고, 상기 선택된 PCUR의 형태를 나타내는 신택스 요소를 시그날링한다.

[0029] 또한, 상기 병렬 코딩 유닛 단위 PCUR의 형태가 정방형으로 선택된 경우, 상기 선택된 정방형 PCUR의 크기를 나타내는 신택스 요소를 시그날링한다.

[0030] 본 발명에 따른 영상 복호화 장치는, 현재 코딩 블록에 대한 블록 파티션 정보를 이용하여, 현재 코딩 블록이 이웃 병렬 처리 블록과 적어도 수평, 수직 또는 대각선 방향 중 어느 하나로 인접한 현재 병렬 처리 블록인지를 확인하고, 상기 현재 병렬 처리 블록에 대해 상기 이웃 병렬 처리 블록과 함께 병렬 복호화하는 복호화기를 포함한다.

[0031] 본 발명에 따른 영상 신호 비트스트림을 포함하는 기록매체에 있어서, 상기 기록매체에 포함된 영상 신호 비트스트림은, 현재 코딩 블록에 대한 블록 파티션 정보를 이용하여, 현재 코딩 블록이 이웃 병렬 처리 블록과 적어도 수평, 수직 또는 대각선 방향 중 어느 하나로 인접한 현재 병렬 처리 블록인지를 결정하는 단계, 및 상기 현재 병렬 처리 블록에 대해 상기 이웃 병렬 처리 블록과 함께 병렬 부호화하는 단계를 포함하는 영상 부호화 방법에 의해 부호화된 것을 특징으로 한다.

[0032] 본 발명에 대하여 위에서 간략하게 요약된 특징들은 후술하는 본 발명의 상세한 설명의 예시적인 양상일 뿐이며, 본 발명의 범위를 제한하는 것은 아니다.

발명의 효과

[0033] 본 발명에 의하면, 효율적으로 부호화/복호화 대상 블록을 분할함으로써, 영상 신호의 부호화/복호화 효율을 증가시킬 수 있다.

[0034] 본 발명에 의하면, 부호화/복호화 대상 블록을 대칭 형태 또는 비대칭 형태의 블록으로 분할함으로써 영상 신호의 부호화/복호화 효율을 증가시킬 수 있다.

[0035] 본 발명에 의하면, 부호화/복호화 대상 블록을 독립적으로 병렬 처리함으로써, 영상 신호의 부호화/복호화 효율을 증가시킬 수 있다.

[0036] 본 발명에서 얻을 수 있는 효과는 이상에서 언급한 효과들로 제한되지 않으며, 언급하지 않은 또 다른 효과들은 아래의 기재로부터 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 명확하게 이해될 수 있을 것이다.

도면의 간단한 설명

[0037] 도 1은 본 발명의 일 실시예에 따른 영상 부호화 장치를 나타낸 블록도이다.

도 2는 본 발명의 일 실시예에 따른 영상 복호화 장치를 나타낸 블록도이다.

도 3은 코딩 블록이 화면 간 예측으로 부호화되었을 때, 코딩 블록에 적용될 수 있는 파티션 모드를 예시한 도면이다.

도 4는 발명이 적용되는 일 실시예로서, 쿼드 트리(Quad tree) 및 바이너리 트리(Binary tree) 분할(partitioning)이 허용되는 파티션 형태를 나타낸 도면이다.

도 5는 본 발명이 적용되는 일 실시예로서, 쿼드 트리 및 바이너리 트리 분할에 기반하여 코딩 블록을 계층적으로 분할하는 일예를 도시한 것이다.

도 6은 본 발명이 적용되는 일 실시예로서, 쿼드 트리 및 대칭형 바이너리 트리 분할에 기반하여 코딩 블록을 계층적으로 분할하는 일예를 도시한 것이다.

도 7은 본 발명이 적용되는 일 실시예로서, 비대칭형 바이너리 트리 분할이 허용되는 파티션 형태를 나타낸 도면이다.

도 8은 본 발명이 적용되는 일 실시예로서, 쿼드 트리 및 대칭형/비대칭형 바이너리 트리 분할에 기반한 코딩 블록의 분할 형태를 예시한 것이다.

도 9는 본 발명이 적용되는 일 실시예로서, 쿼드 트리 및 바이너리 트리 분할에 기반한 코딩 블록 분할 방법에 대한 흐름도이다.

도 10은 본 발명이 적용되는 일 실시예로서, 쿼드 트리 및 바이너리 트리 분할이 적용되는 네트워크 추상화 계층 (NAL)에 포함되는 신택스 요소(syntax element)를 예를 들어 도시한 것이다.

도 11은 본 발명이 적용되는 다른 실시예로서, 비대칭형 쿼드 트리 분할이 허용되는 파티션 형태를 나타낸 도면이다.

도 12는 본 발명이 적용되는 다른 실시예로서, 비대칭형 쿼드 트리 분할에 기반한 코딩 블록 분할 방법에 대한 흐름도이다.

도 13은 본 발명이 적용되는 다른 실시예로서, 비대칭형 쿼드 트리 분할이 적용되는 네트워크 추상화 계층 (NAL)에 포함되는 신택스 요소(syntax element)를 예를 들어 도시한 것이다.

도 14는 본 발명이 적용되는 또 다른 실시예로서, 쿼드 트리 및 트리플 트리 분할이 허용되는 파티션 형태를 나타낸 도면이다.

도 15는 본 발명이 적용되는 또 다른 실시예로서, 쿼드 트리 및 트리플 트리 분할 에 기반한 코딩 블록 분할 방법에 대한 흐름도이다.

도 16은 본 발명이 적용되는 또 다른 실시예로서, 쿼드 트리 및 트리플 트리 분할이 적용되는 네트워크 추상화 계층 (NAL)에 포함되는 신택스 요소(syntax element)를 예를 들어 도시한 것이다.

도 17은 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할이 허용되는 기본 파티션 형태를 나타낸 도면이다.

도 18은 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할이 허용되는 확장된 파티션 형태를 나타낸 도면이다.

도 19는 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할에 기반한 코딩 블록 분할 방법에 대한 흐름도이다.

도 20은 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할에 의해 분할된 제1타입 코딩 블록의 병렬 처리 방법을 설명하기 위해 도시한 것이다.

도 21은 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할에 의해 분할된 제2타입 코딩 블록의 병렬 처리 방법을 설명하기 위해 도시한 것이다.

도 22는 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할에 의해 분할된 병렬 처리 가능한 코딩 블록을 설명하기 위해 도시한 것이다.

도 23은 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할에 의해 분할된 병렬 처리 가능한 병렬 코딩 유닛 단위 (PCUR: Parallel Coding Unit processing Region)를 설명하기 위해 도시한 것이다.

도 24는 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할에 의해 분할된 코딩 블록의 병렬 처리 부호화 방법을 설명하기 위해 도시한 것이다.

도 25는 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할에 의해 분할된 코딩 블록의 병렬 처리 복호화 방법을 설명하기 위해 도시한 것이다.

도 26 및 도 27은 본 발명이 적용되는 또 다른 실시예로서, 제1타입 코딩 블록의 병렬 처리를 위해 참조 샘플을 획득하는 방법을 설명하기 위해 도시한 것이다.

도 28 및 도 29는 본 발명이 적용되는 또 다른 실시예로서, 제2타입 코딩 블록의 병렬 처리를 위해 참조 샘플을 획득하는 방법을 설명하기 위해 도시한 것이다.

도 30은 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할에 분할된 병렬 처리 코딩 블록이 적용되는 네트워크 추상화 계층 (NAL)에 포함되는 신택스 요소(syntax element)를 예를 들어 도시한 것이다.

발명을 실시하기 위한 구체적인 내용

[0038] 본 발명은 다양한 변경을 가할 수 있고 여러 가지 실시예를 가질 수 있는 바, 특정 실시예들을 도면에 예시하고 상세한 설명에 상세하게 설명하고자 한다. 그러나, 이는 본 발명을 특정한 실시 형태에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다. 각 도면을 설명하면서 유사한 참조부호를 유사한 구성요소에 대해 사용하였다.

[0039] 제1, 제2 등의 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 상기 구성요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 사용된다. 예를 들어, 본 발명의 권리 범위를 벗어나지 않으면서 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소도 제1 구성요소로 명명될 수 있다. 및/또는 이라는 용어는 복수의 관련된 기재된 항목들의 조합 또는 복수의 관련된 기재된 항목들 중의 어느 항목을 포함한다.

[0040] 본 출원에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 출원에서, "포함하다" 또는 "가지다" 등의 용어는 명세서상에 기재된 특징, 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0041] 또한, 본 출원에서 사용한 “유닛(unit)”은 “블록(block)”으로 대체할 수 있으며, 따라서, 본 명세서에서 “코딩 트리 유닛”과 “코딩 트리 블록”, “코딩 유닛”과 “코딩 블록”, “예측 유닛”과 “예측 블록”, “변환 유닛”과 “변환 블록”은 각각 동일한 의미로 해석할 수 있다.

[0042] 이하, 첨부한 도면들을 참조하여, 본 발명의 바람직한 실시예를 보다 상세하게 설명하고자 한다. 이하, 도면상의 동일한 구성요소에 대해서는 동일한 참조부호를 사용하고 동일한 구성요소에 대해서 중복된 설명은 생략한다.

[0043] 도 1은 본 발명의 일실시예에 따른 영상 부호화 장치를 나타낸 블록도이다.

[0044] 도 1을 참조하면, 영상 부호화 장치(100)는 픽쳐 분할부(110), 예측부(120, 125), 변환부(130), 양자화부(135), 재정렬부(160), 엔트로피 부호화부(165), 역양자화부(140), 역변환부(145), 필터부(150) 및 메모리(155)를 포함할 수 있다.

[0045] 도 1에 나타난 각 구성부들은 영상 부호화 장치에서 서로 다른 특징적인 기능들을 나타내기 위해 독립적으로 도시한 것으로, 각 구성부들이 분리된 하드웨어나 하나의 소프트웨어 구성단위로 이루어짐을 의미하지 않는다. 즉, 각 구성부는 설명의 편의상 각각의 구성부로 나열하여 포함한 것으로 각 구성부 중 적어도 두 개의 구성부가 합쳐져 하나의 구성부로 이루어지거나, 하나의 구성부가 복수개의 구성부로 나뉘어져 기능을 수행할 수 있고 이러한 각 구성부의 통합된 실시예 및 분리된 실시예도 본 발명의 본질에서 벗어나지 않는 한 본 발명의 권리범위에 포함된다.

[0046] 또한, 일부의 구성 요소는 본 발명에서 본질적인 기능을 수행하는 필수적인 구성 요소는 아니고 단지 성능을 향상시키기 위한 선택적 구성 요소일 수 있다. 본 발명은 단지 성능 향상을 위해 사용되는 구성 요소를 제외한 본 발명의 본질을 구현하는데 필수적인 구성부만을 포함하여 구현될 수 있고, 단지 성능 향상을 위해 사용되는 선택적 구성 요소를 제외한 필수 구성 요소만을 포함한 구조도 본 발명의 권리범위에 포함된다.

[0047] 픽쳐 분할부(110)는 입력된 픽쳐를 적어도 하나의 처리 단위로 분할할 수 있다. 이때, 처리 단위는 예측 단위(Prediction Unit: PU)일 수도 있고, 변환 단위(Transform Unit: TU)일 수도 있으며, 부호화 단위(Coding Unit: CU)일 수도 있다. 픽쳐 분할부(110)에서는 하나의 픽쳐에 대해 복수의 부호화 단위, 예측 단위 및 변환 단위의 조합으로 분할하고 소정의 기준(예를 들어, 비용 함수)으로 하나의 부호화 단위, 예측 단위 및 변환 단위 조합을 선택하여 픽쳐를 부호화 할 수 있다.

[0048] 예를 들어, 하나의 픽쳐는 복수개의 부호화 단위로 분할될 수 있다. 픽쳐에서 부호화 단위를 분할하기 위해서는 쿼드 트리 구조(Quad Tree Structure)와 같은 재귀적인 트리 구조를 사용할 수 있는데 하나의 영상 또는 최대 크기 부호화 단위(largest coding unit)를 루트로 하여 다른 부호화 단위로 분할되는 부호화 유닛은 분할된 부

호화 단위의 개수만큼의 자식 노드를 가지고 분할될 수 있다. 일정한 제한에 따라 더 이상 분할되지 않는 부호화 단위는 리프 노드가 된다. 즉, 하나의 코딩 유닛에 대하여 정방형 분할만이 가능하다고 가정하는 경우, 하나의 부호화 단위는 최대 4개의 다른 부호화 단위로 분할될 수 있다.

[0049] 이하, 본 발명의 실시예에서는 부호화 단위는 부호화를 수행하는 단위의 의미로 사용할 수도 있고, 복호화를 수행하는 단위의 의미로 사용할 수도 있다.

[0050] 예측 단위는 하나의 부호화 단위 내에서 동일한 크기의 적어도 하나의 정사각형 또는 직사각형 등의 형태를 가지고 분할된 것일 수도 있고, 하나의 부호화 단위 내에서 분할된 예측 단위 중 어느 하나의 예측 단위가 다른 하나의 예측 단위와 상이한 형태 및/또는 크기를 가지도록 분할된 것일 수도 있다.

[0051] 부호화 단위를 기초로 인트라 예측을 수행하는 예측 단위를 생성시 최소 부호화 단위가 아닌 경우, 복수의 예측 단위 NxN 으로 분할하지 않고 인트라 예측을 수행할 수 있다.

[0052] 예측부(120, 125)는 인터 예측을 수행하는 인터 예측부(120)와 인트라 예측을 수행하는 인트라 예측부(125)를 포함할 수 있다. 예측 단위에 대해 인터 예측을 사용할 것인지 또는 인트라 예측을 수행할 것인지를 결정하고, 각 예측 방법에 따른 구체적인 정보(예컨대, 인트라 예측 모드, 모션 벡터, 참조 픽쳐 등)를 결정할 수 있다. 이때, 예측이 수행되는 처리 단위와 예측 방법 및 구체적인 내용이 정해지는 처리 단위는 다를 수 있다. 예컨대, 예측의 방법과 예측 모드 등은 예측 단위로 결정되고, 예측의 수행은 변환 단위로 수행될 수도 있다. 생성된 예측 블록과 원본 블록 사이의 잔차값(잔차 블록)은 변환부(130)로 입력될 수 있다. 또한, 예측을 위해 사용한 예측 모드 정보, 모션 벡터 정보 등은 잔차값과 함께 엔트로피 부호화부(165)에서 부호화되어 복호화기에 전달될 수 있다. 특정한 부호화 모드를 사용할 경우, 예측부(120, 125)를 통해 예측 블록을 생성하지 않고, 원본 블록을 그대로 부호화하여 복호화부에 전송하는 것도 가능하다.

[0053] 인터 예측부(120)는 현재 픽쳐의 이전 픽쳐 또는 이후 픽쳐 중 적어도 하나의 픽쳐의 정보를 기초로 예측 단위를 예측할 수도 있고, 경우에 따라서는 현재 픽쳐 내의 부호화가 완료된 일부 영역의 정보를 기초로 예측 단위를 예측할 수도 있다. 인터 예측부(120)는 참조 픽쳐 보간부, 모션 예측부, 움직임 보상부를 포함할 수 있다.

[0054] 참조 픽쳐 보간부에서는 메모리(155)로부터 참조 픽쳐 정보를 제공받고 참조 픽쳐에서 정수 화소 이하의 화소 정보를 생성할 수 있다. 휘도 화소의 경우, 1/4 화소 단위로 정수 화소 이하의 화소 정보를 생성하기 위해 필터 계수를 달리하는 DCT 기반의 8탭 보간 필터(DCT-based Interpolation Filter)가 사용될 수 있다. 색차 신호의 경우 1/8 화소 단위로 정수 화소 이하의 화소 정보를 생성하기 위해 필터 계수를 달리하는 DCT 기반의 4탭 보간 필터(DCT-based Interpolation Filter)가 사용될 수 있다.

[0055] 모션 예측부는 참조 픽쳐 보간부에 의해 보간된 참조 픽쳐를 기초로 모션 예측을 수행할 수 있다. 모션 벡터를 산출하기 위한 방법으로 FBMA(Full search-based Block Matching Algorithm), TSS(Three Step Search), NTS(New Three-Step Search Algorithm) 등 다양한 방법이 사용될 수 있다. 모션 벡터는 보간된 화소를 기초로 1/2 또는 1/4 화소 단위의 모션 벡터값을 가질 수 있다. 모션 예측부에서는 모션 예측 방법을 다르게 하여 현재 예측 단위를 예측할 수 있다. 모션 예측 방법으로 스킵(Skip) 방법, 머지(Merge) 방법, AMVP(Advanced Motion Vector Prediction) 방법, 인트라 블록 카피(Intra Block Copy) 방법 등 다양한 방법이 사용될 수 있다.

[0056] 인트라 예측부(125)는 현재 픽쳐 내의 화소 정보인 현재 블록 주변의 참조 픽셀 정보를 기초로 예측 단위를 생성할 수 있다. 현재 예측 단위의 주변 블록이 인터 예측을 수행한 블록이어서, 참조 픽셀이 인터 예측을 수행한 픽셀일 경우, 인터 예측을 수행한 블록에 포함되는 참조 픽셀을 주변의 인트라 예측을 수행한 블록의 참조 픽셀 정보로 대체하여 사용할 수 있다. 즉, 참조 픽셀이 가용하지 않는 경우, 가용하지 않은 참조 픽셀 정보를 가용한 참조 픽셀 중 적어도 하나의 참조 픽셀로 대체하여 사용할 수 있다.

[0057] 인트라 예측에서 예측 모드는 참조 픽셀 정보를 예측 방향에 따라 사용하는 방향성 예측 모드와 예측을 수행시 방향성 정보를 사용하지 않는 비방향성 모드를 가질 수 있다. 휘도 정보를 예측하기 위한 모드와 색차 정보를 예측하기 위한 모드가 상이할 수 있고, 색차 정보를 예측하기 위해 휘도 정보를 예측하기 위해 사용된 인트라 예측 모드 정보 또는 예측된 휘도 신호 정보를 활용할 수 있다.

[0058] 인트라 예측을 수행할 때 예측 단위의 크기와 변환 단위의 크기가 동일할 경우, 예측 단위의 좌측에 존재하는 픽셀, 좌측 상단에 존재하는 픽셀, 상단에 존재하는 픽셀을 기초로 예측 단위에 대한 인트라 예측을 수행할 수 있다. 그러나 인트라 예측을 수행할 때 예측 단위의 크기와 변환 단위의 크기가 상이할 경우, 변환 단위를 기초로 한 참조 픽셀을 이용하여 인트라 예측을 수행할 수 있다. 또한, 최소 부호화 단위에 대해서만 NxN 분할을 사

용하는 인트라 예측을 사용할 수 있다.

[0059] 인트라 예측 방법은 예측 모드에 따라 참조 화소에 AIS(Adaptive Intra Smoothing) 필터를 적용한 후 예측 블록을 생성할 수 있다. 참조 화소에 적용되는 AIS 필터의 종류는 상이할 수 있다. 인트라 예측 방법을 수행하기 위해 현재 예측 단위의 인트라 예측 모드는 현재 예측 단위의 주변에 존재하는 예측 단위의 인트라 예측 모드로부터 예측할 수 있다. 주변 예측 단위로부터 예측된 모드 정보를 이용하여 현재 예측 단위의 예측 모드를 예측하는 경우, 현재 예측 단위와 주변 예측 단위의 인트라 예측 모드가 동일하면 소정의 플래그 정보를 이용하여 현재 예측 단위와 주변 예측 단위의 예측 모드가 동일하다는 정보를 전송할 수 있고, 만약 현재 예측 단위와 주변 예측 단위의 예측 모드가 상이하면 엔트로피 부호화를 수행하여 현재 블록의 예측 모드 정보를 부호화할 수 있다.

[0060] 또한, 예측부(120, 125)에서 생성된 예측 단위를 기초로 예측을 수행한 예측 단위와 예측 단위의 원본 블록과 차이값인 잔차값(Residual) 정보를 포함하는 잔차 블록이 생성될 수 있다. 생성된 잔차 블록은 변환부(130)로 입력될 수 있다.

[0061] 변환부(130)에서는 원본 블록과 예측부(120, 125)를 통해 생성된 예측 단위의 잔차값(residual)정보를 포함한 잔차 블록을 DCT(Discrete Cosine Transform), DST(Discrete Sine Transform), KLT와 같은 변환 방법을 사용하여 변환시킬 수 있다. 잔차 블록을 변환하기 위해 DCT를 적용할지, DST를 적용할지 또는 KLT를 적용할지는 잔차 블록을 생성하기 위해 사용된 예측 단위의 인트라 예측 모드 정보를 기초로 결정할 수 있다.

[0062] 양자화부(135)는 변환부(130)에서 주파수 영역으로 변환된 값들을 양자화할 수 있다. 블록에 따라 또는 영상의 중요도에 따라 양자화 계수는 변할 수 있다. 양자화부(135)에서 산출된 값은 역양자화부(140)와 재정렬부(160)에 제공될 수 있다.

[0063] 재정렬부(160)는 양자화된 잔차값에 대해 계수값의 재정렬을 수행할 수 있다.

[0064] 재정렬부(160)는 계수 스캐닝(Coefficient Scanning) 방법을 통해 2차원의 블록 형태 계수를 1차원의 벡터 형태로 변경할 수 있다. 예를 들어, 재정렬부(160)에서는 지그-재그 스캔(Zig-Zag Scan)방법을 이용하여 DC 계수부터 고주파수 영역의 계수까지 스캔하여 1차원 벡터 형태로 변경시킬 수 있다. 변환 단위의 크기 및 인트라 예측 모드에 따라 지그-재그 스캔 대신 2차원의 블록 형태 계수를 열 방향으로 스캔하는 수직 스캔, 2차원의 블록 형태 계수를 행 방향으로 스캔하는 수평 스캔이 사용될 수도 있다. 즉, 변환 단위의 크기 및 인트라 예측 모드에 따라 지그-재그 스캔, 수직 방향 스캔 및 수평 방향 스캔 중 어떠한 스캔 방법이 사용될지 여부를 결정할 수 있다.

[0065] 엔트로피 부호화부(165)는 재정렬부(160)에 의해 산출된 값들을 기초로 엔트로피 부호화를 수행할 수 있다. 엔트로피 부호화는 예를 들어, 지수 골롬(Exponential Golomb), CAVLC(Context-Adaptive Variable Length Coding), CABAC(Context-Adaptive Binary Arithmetic Coding)과 같은 다양한 부호화 방법을 사용할 수 있다.

[0066] 엔트로피 부호화부(165)는 재정렬부(160) 및 예측부(120, 125)로부터 부호화 단위의 잔차값 계수 정보 및 블록 타입 정보, 예측 모드 정보, 분할 단위 정보, 예측 단위 정보 및 전송 단위 정보, 모션 벡터 정보, 참조 프레임 정보, 블록의 보간 정보, 필터링 정보 등 다양한 정보를 부호화할 수 있다.

[0067] 엔트로피 부호화부(165)에서는 재정렬부(160)에서 입력된 부호화 단위의 계수값을 엔트로피 부호화할 수 있다.

[0068] 역양자화부(140) 및 역변환부(145)에서는 양자화부(135)에서 양자화된 값들을 역양자화하고 변환부(130)에서 변환된 값들을 역변환한다. 역양자화부(140) 및 역변환부(145)에서 생성된 잔차값(Residual)은 예측부(120, 125)에 포함된 움직임 추정부, 움직임 보상부 및 인트라 예측부를 통해서 예측된 예측 단위와 합쳐져 복원 블록(Reconstructed Block)을 생성할 수 있다.

[0069] 필터부(150)는 디블록킹 필터, 오프셋 보정부, ALF(Adaptive Loop Filter)중 적어도 하나를 포함할 수 있다.

[0070] 디블록킹 필터는 복원된 픽처에서 블록간의 경계로 인해 생긴 블록 왜곡을 제거할 수 있다. 디블록킹을 수행할지 여부를 판단하기 위해 블록에 포함된 몇 개의 열 또는 행에 포함된 픽셀을 기초로 현재 블록에 디블록킹 필터 적용할지 여부를 판단할 수 있다. 블록에 디블록킹 필터를 적용하는 경우 필요한 디블록킹 필터링 강도에 따라 강한 필터(Strong Filter) 또는 약한 필터(Weak Filter)를 적용할 수 있다. 또한 디블록킹 필터를 적용함에 있어 수직 필터링 및 수평 필터링 수행시 수평 방향 필터링 및 수직 방향 필터링이 병행 처리되도록 할 수 있다.

[0071] 오프셋 보정부는 디블록킹을 수행한 영상에 대해 픽셀 단위로 원본 영상과의 오프셋을 보정할 수 있다. 특정 픽처에 대한 오프셋 보정을 수행하기 위해 영상에 포함된 픽셀을 일정한 수의 영역으로 구분한 후 오프셋을 수행할 영역을 결정하고 해당 영역에 오프셋을 적용하는 방법 또는 각 픽셀의 에지 정보를 고려하여 오프셋을 적용하는 방법을 사용할 수 있다.

[0072] ALF(Adaptive Loop Filtering)는 필터링한 복원 영상과 원래의 영상을 비교한 값을 기초로 수행될 수 있다. 영상에 포함된 픽셀을 소정의 그룹으로 나눈 후 해당 그룹에 적용될 하나의 필터를 결정하여 그룹마다 차별적으로 필터링을 수행할 수 있다. ALF를 적용할지 여부에 관련된 정보는 휘도 신호는 부호화 단위(Coding Unit, CU) 별로 전송될 수 있고, 각각의 블록에 따라 적용될 ALF 필터의 모양 및 필터 계수는 달라질 수 있다. 또한, 적용 대상 블록의 특성에 상관없이 동일한 형태(고정된 형태)의 ALF 필터가 적용될 수도 있다.

[0073] 메모리(155)는 필터부(150)를 통해 산출된 복원 블록 또는 픽처를 저장할 수 있고, 저장된 복원 블록 또는 픽처는 인터 예측을 수행 시 예측부(120, 125)에 제공될 수 있다.

[0074] 도 2는 본 발명의 일실시예에 따른 영상 복호화 장치를 나타낸 블록도이다.

[0075] 도 2를 참조하면, 영상 복호화기(200)는 엔트로피 복호화부(210), 재정렬부(215), 역양자화부(220), 역변환부(225), 예측부(230, 235), 필터부(240), 메모리(245)가 포함될 수 있다.

[0076] 영상 부호화기에서 영상 비트스트림이 입력된 경우, 입력된 비트스트림은 영상 부호화기와 반대의 절차로 복호화될 수 있다.

[0077] 엔트로피 복호화부(210)는 영상 부호화기의 엔트로피 부호화부에서 엔트로피 부호화를 수행한 것과 반대의 절차로 엔트로피 복호화를 수행할 수 있다. 예를 들어, 영상 부호화기에서 수행된 방법에 대응하여 지수 골롬(Exponential Golomb), CAVLC(Context-Adaptive Variable Length Coding), CABAC(Context-Adaptive Binary Arithmetic Coding)과 같은 다양한 방법이 적용될 수 있다.

[0078] 엔트로피 복호화부(210)에서는 부호화기에서 수행된 인트라 예측 및 인터 예측에 관련된 정보를 복호화할 수 있다.

[0079] 재정렬부(215)는 엔트로피 복호화부(210)에서 엔트로피 복호화된 비트스트림을 부호화부에서 재정렬한 방법을 기초로 재정렬을 수행할 수 있다. 1차원 벡터 형태로 표현된 계수들을 다시 2차원의 블록 형태의 계수로 복원하여 재정렬할 수 있다. 재정렬부(215)에서는 부호화부에서 수행된 계수 스캐닝에 관련된 정보를 제공받고 해당 부호화부에서 수행된 스캐닝 순서에 기초하여 역으로 스캐닝하는 방법을 통해 재정렬을 수행할 수 있다.

[0080] 역양자화부(220)는 부호화기에서 제공된 양자화 파라미터와 재정렬된 블록의 계수값을 기초로 역양자화를 수행할 수 있다.

[0081] 역변환부(225)는 영상 부호화기에서 수행한 양자화 결과에 대해 변환부에서 수행한 변환 즉, DCT, DST, 및 KLT에 대해 역변환 즉, 역 DCT, 역 DST 및 역 KLT를 수행할 수 있다. 역변환은 영상 부호화기에서 결정된 전송 단위를 기초로 수행될 수 있다. 영상 복호화기의 역변환부(225)에서는 예측 방법, 현재 블록의 크기 및 예측 방향 등 복수의 정보에 따라 변환 기법(예를 들어, DCT, DST, KLT)이 선택적으로 수행될 수 있다.

[0082] 예측부(230, 235)는 엔트로피 복호화부(210)에서 제공된 예측 블록 생성 관련 정보와 메모리(245)에서 제공된 이전에 복호화된 블록 또는 픽처 정보를 기초로 예측 블록을 생성할 수 있다.

[0083] 전술한 바와 같이 영상 부호화기에서의 동작과 동일하게 인트라 예측을 수행시 예측 단위의 크기와 변환 단위의 크기가 동일할 경우, 예측 단위의 좌측에 존재하는 픽셀, 좌측 상단에 존재하는 픽셀, 상단에 존재하는 픽셀을 기초로 예측 단위에 대한 인트라 예측을 수행하지만, 인트라 예측을 수행시 예측 단위의 크기와 변환 단위의 크기가 상이할 경우, 변환 단위를 기초로 한 참조 픽셀을 이용하여 인트라 예측을 수행할 수 있다. 또한, 최소 부호화 단위에 대해서만 NxN 분할을 사용하는 인트라 예측을 사용할 수도 있다.

[0084] 예측부(230, 235)는 예측 단위 판별부, 인터 예측부 및 인트라 예측부를 포함할 수 있다. 예측 단위 판별부는 엔트로피 복호화부(210)에서 입력되는 예측 단위 정보, 인트라 예측 방법의 예측 모드 정보, 인터 예측 방법의 모션 예측 관련 정보 등 다양한 정보를 입력 받고 현재 부호화 단위에서 예측 단위를 구분하고, 예측 단위가 인터 예측을 수행하는지 아니면 인트라 예측을 수행하는지 여부를 판별할 수 있다. 인터 예측부(230)는 영상 부호화기에서 제공된 현재 예측 단위의 인터 예측에 필요한 정보를 이용해 현재 예측 단위가 포함된 현재 픽처의 이전 픽처 또는 이후 픽처 중 적어도 하나의 픽처에 포함된 정보를 기초로 현재 예측 단위에 대한 인터 예측을 수

행할 수 있다. 또는, 현재 예측 단위가 포함된 현재 픽쳐 내에서 기-복원된 일부 영역의 정보를 기초로 인터 예측을 수행할 수도 있다.

[0085] 인터 예측을 수행하기 위해 부호화 단위를 기준으로 해당 부호화 단위에 포함된 예측 단위의 모션 예측 방법이 스킵 모드(Skip Mode), 머지 모드(Merge 모드), AMVP 모드(AMVP Mode), 인트라 블록 카피 모드 중 어떠한 방법인지 여부를 판단할 수 있다.

[0086] 인트라 예측부(235)는 현재 픽쳐 내의 화소 정보를 기초로 예측 블록을 생성할 수 있다. 예측 단위가 인트라 예측을 수행한 예측 단위인 경우, 영상 부호화기에서 제공된 예측 단위의 인트라 예측 모드 정보를 기초로 인트라 예측을 수행할 수 있다. 인트라 예측부(235)에는 AIS(Adaptive Intra Smoothing) 필터, 참조 화소 보간부, DC 필터를 포함할 수 있다. AIS 필터는 현재 블록의 참조 화소에 필터링을 수행하는 부분으로써 현재 예측 단위의 예측 모드에 따라 필터의 적용 여부를 결정하여 적용할 수 있다. 영상 부호화기에서 제공된 예측 단위의 예측 모드 및 AIS 필터 정보를 이용하여 현재 블록의 참조 화소에 AIS 필터링을 수행할 수 있다. 현재 블록의 예측 모드가 AIS 필터링을 수행하지 않는 모드일 경우, AIS 필터는 적용되지 않을 수 있다.

[0087] 참조 화소 보간부는 예측 단위의 예측 모드가 참조 화소를 보간한 화소값을 기초로 인트라 예측을 수행하는 예측 단위일 경우, 참조 화소를 보간하여 정수값 이하의 화소 단위의 참조 화소를 생성할 수 있다. 현재 예측 단위의 예측 모드가 참조 화소를 보간하지 않고 예측 블록을 생성하는 예측 모드일 경우 참조 화소는 보간되지 않을 수 있다. DC 필터는 현재 블록의 예측 모드가 DC 모드일 경우 필터링을 통해서 예측 블록을 생성할 수 있다.

[0088] 복원된 블록 또는 픽쳐는 필터부(240)로 제공될 수 있다. 필터부(240)는 디블록킹 필터, 오프셋 보정부, ALF를 포함할 수 있다.

[0089] 영상 부호화기로부터 해당 블록 또는 픽쳐에 디블록킹 필터를 적용하였는지 여부에 대한 정보 및 디블록킹 필터를 적용하였을 경우, 강한 필터를 적용하였는지 또는 약한 필터를 적용하였는지에 대한 정보를 제공받을 수 있다. 영상 복호화기의 디블록킹 필터에서는 영상 부호화기에서 제공된 디블록킹 필터 관련 정보를 제공받고 영상 복호화기에서 해당 블록에 대한 디블록킹 필터링을 수행할 수 있다.

[0090] 오프셋 보정부는 부호화시 영상에 적용된 오프셋 보정의 종류 및 오프셋 값 정보 등을 기초로 복원된 영상에 오프셋 보정을 수행할 수 있다.

[0091] ALF는 부호화기로부터 제공된 ALF 적용 여부 정보, ALF 계수 정보 등을 기초로 부호화 단위에 적용될 수 있다. 이러한 ALF 정보는 특정한 파라메터 셋에 포함되어 제공될 수 있다.

[0092] 메모리(245)는 복원된 픽쳐 또는 블록을 저장하여 참조 픽쳐 또는 참조 블록으로 사용할 수 있도록 할 수 있고 또한 복원된 픽쳐를 출력부로 제공할 수 있다.

[0093] 전술한 바와 같이 이하, 본 발명의 실시예에서는 설명의 편의상 코딩 유닛(Coding Unit)을 부호화 단위라는 용어로 사용하지만, 부호화뿐만 아니라 복호화를 수행하는 단위가 될 수도 있다.

[0094] 또한, 현재 블록은, 부호화/복호화 대상 블록을 나타내는 것으로,부호화/복호화 단계에 따라, 코딩 트리 블록(또는 코딩 트리 유닛), 부호화 블록(또는 부호화 유닛), 변환 블록(또는 변환 유닛) 또는 예측 블록(또는 예측 유닛) 등을 나타내는 것일 수 있다. 본 명세서에서, '유닛'은 특정 부호화/복호화 프로세스를 수행하기 위한 기본 단위를 나타내고, '블록'은 소정 크기의 샘플 어레이를 나타낼 수 있다. 별도의 구분이 없는 한, '블록'과 '유닛'은 동등한 의미로 사용될 수 있다. 예컨대, 후술되는 실시예에서, 부호화 블록(코딩 블록) 및 부호화 유닛(코딩 유닛)은 상호 동등한 의미인 것으로 이해될 수 있다.

[0095] 하나의 픽쳐는 정방형 또는 비정방형의 기본 블록으로 분할되어 부호화/복호화될 수 있다. 이때, 기본 블록은, 코딩 트리 유닛(Coding Tree Unit)이라 호칭될 수 있다. 코딩 트리 유닛은, 시퀀스 또는 슬라이스에서 허용하는 가장 큰 크기의 코딩 유닛으로 정의될 수도 있다. 코딩 트리 유닛이 정방형 또는 비정방형인지 여부 또는 코딩 트리 유닛의 크기와 관련한 정보는 시퀀스 파라미터 셋트, 픽쳐 파라미터 셋트 또는 슬라이스 헤더 등을 통해 시그널링될 수 있다. 코딩 트리 유닛은 더 작은 크기의 파티션으로 분할될 수 있다. 이때, 코딩 트리 유닛을 분할함으로써 생성된 파티션을 뎁스 1이라 할 경우, 뎁스 1인 파티션을 분할함으로써 생성된 파티션은 뎁스 2로 정의될 수 있다. 즉, 코딩 트리 유닛 내 뎁스 k인 파티션을 분할함으로써 생성된 파티션은 뎁스 k+1을 갖는 것으로 정의될 수 있다.

[0096] 도 3은 코딩 블록이 화면 내 예측 또는 화면 간 예측으로 부호화되었을 때, 코딩 블록에 적용될 수 있는 파티션 모드를 예시한 도면이다.코딩 트리 유닛이 분할됨에 따라 생성된 임의 크기의 파티션을 코딩 유닛이라 정의할

수 있다. 예를 들어, 도 3 (a)는 코딩 유닛이 2Nx2N 크기을 도시하였다. 코딩 유닛은 재귀적으로 분할되거나, 예측, 양자화, 변환 또는 인루프 필터링 등을 수행하기 위한 기본 단위로 분할될 수 있다. 일 예로, 코딩 유닛이 분할됨에 따라 생성된 임의 크기의 파티션은 코딩 유닛으로 정의되거나, 예측, 양자화, 변환 또는 인루프 필터링 등을 수행하기 위한 기본 단위인 변환 유닛(TU: Transform Unit) 또는 예측 유닛(PU: Prediction Unit)으로 정의될 수 있다.

[0097] 또는, 코딩 블록이 결정되면, 코딩 블록의 예측 분할을 통해 코딩 블록과 동일한 크기 또는 코딩 블록보다 작은 크기를 갖는 예측 블록(Prediction Block)을 결정할 수 있다. 코딩 블록의 예측 분할은 코딩 블록의 분할 형태를 나타내는 파티션 모드(Part_mode)에 의해 수행될 수 있다. 예측 블록의 크기 또는 형태는 코딩 블록의 파티션 모드에 따라 결정될 수 있다. 코딩 블록의 분할 형태는 파티션 후보 중 어느 하나를 특정하는 정보를 통해 결정될 수 있다. 이때, 코딩 블록이 이용할 수 있는 파티션 후보에는 코딩 블록의 크기, 형태 또는 부호화 모드 등에 따라 비대칭 파티션 형태(예컨대, nLx2N, nRx2N, 2NxnU, 2NxnD)가 포함될 수 있다. 일 예로, 코딩 블록이 이용할 수 있는 파티션 후보는 현재 블록의 부호화 모드에 따라 결정될 수 있다. 예를 들어, 코딩 블록이 화면 간 예측으로 부호화된 경우, 코딩 블록에는 도 3 (b)에 도시된 예에서와 같이, 8개의 파티션 모드 중 어느 하나가 적용될 수 있다. 반면, 코딩 블록이 화면 내 예측으로 부호화된 경우, 코딩 블록에는 도 3 (b)의 8개 파티션 모드중 PART_2Nx2N 또는 PART_NxN 이 적용될 수 있다.

[0098] PART_NxN은 코딩 블록이 최소 크기를 갖는 경우 적용될 수 있다. 여기서, 코딩 블록의 최소 크기는 부호화기 및 복호화기에서 기 정의된 것일 수 있다. 또는, 코딩 블록의 최소 크기에 관한 정보는 비트스트림을 통해 시그널링될 수도 있다. 일 예로, 코딩 블록의 최소 크기는 슬라이스 헤더를 통해 시그널링되고, 이에 따라, 슬라이스별로 코딩 블록의 최소 크기가 정의될 수 있다.

[0099] 다른 예로, 코딩 블록이 이용할 수 있는 파티션 후보는 코딩 블록의 크기 또는 형태 중 적어도 하나에 따라 상이하게 결정될 수도 있다. 일 예로, 코딩 블록이 이용할 수 있는 파티션 후보의 개수 또는 종류는 코딩 블록의 크기 또는 형태 중 적어도 하나에 따라 상이하게 결정될 수 있다.

[0100] 또는, 코딩 블록이 이용할 수 있는 파티션 후보들 중 비대칭 파티션 후보들의 종류 또는 개수를 코딩 블록의 크기 또는 형태에 따라 제한할 수도 있다. 일 예로, 코딩 블록이 이용할 수 있는 비대칭 파티션 후보의 개수 또는 종류는 코딩 블록의 크기 또는 형태 중 적어도 하나에 따라 상이하게 결정될 수 있다.

[0101] 일반적으로, 예측 블록의 크기는 64x64 부터 4x4의 크기를 가질 수 있다. 단, 코딩 블록이 화면 간 예측으로 부호화된 경우, 움직임 보상을 수행할 때, 메모리 대역폭(memory bandwidth)을 줄이기 위해, 예측 블록이 4x4 크기를 갖지 않도록 할 수 있다.

[0102] 파티션 모드를 이용하여, 코딩 블록을 재귀적으로 분할하는 것도 가능하다. 즉, 파티션 인덱스가 지시하는 파티션 모드에 따라 코딩 블록을 분할할 수 있고, 코딩 블록이 분할됨에 따라 생성된 각 파티션이 코딩 블록으로 정의될 수 있다.

[0103] 이하, 코딩 유닛을 재귀적으로 분할하는 방법에 대해 보다 상세히 설명하기로 한다. 설명의 편의를 위해, 이하, 코딩 트리 유닛도 코딩 유닛의 범주에 포함되는 것으로 가정 한다. 즉, 후술되는 실시예에서, 코딩 유닛은, 코딩 트리 유닛을 가리키거나, 코딩 트리 유닛이 분할됨에 따라 생성되는 코딩 유닛을 의미할 수 있다. 또한, 코딩 블록이 재귀적으로 분할되는 경우, 코딩 블록이 분할됨에 따라 생성되는 '파티션'은 '코딩 블록'을 의미하는 것으로 이해될 수 있다.

[0104] 코딩 유닛은 적어도 하나의 라인에 의해 분할될 수 있다. 이때, 코딩 유닛을 분할하는 라인은 소정의 각도를 가질 수도 있다. 여기서, 소정의 각도는, 0도 내지 360도 범위 내의 값일 수 있다. 예컨대, 0도 라인은, 수평 라인, 90도 라인은 수직 라인을 의미하고, 45도 또는 135도 라인은 대각선 라인을 의미할 수 있다.

[0105] 코딩 유닛이 복수의 라인에 의해 분할되는 경우, 복수의 라인은 모두 동일한 각도를 가질 수 있다. 또는, 복수의 라인 중 적어도 하나는 다른 라인과 상이한 각도를 가질 수도 있다. 또는, 코딩 트리 유닛 또는 코딩 유닛을 분할하는 복수의 라인은 기 정의된 각도 차(예컨대, 90도)를 갖도록 설정될 수도 있다.

[0106] 코딩 트리 유닛 또는 코딩 유닛을 분할하는 라인에 관한 정보는, 파티션 모드로 정의되어 부호화될 수 있다. 또는, 라인의 개수, 방향, 각도, 블록 내 라인의 위치 등에 대한 정보가 부호화될 수도 있다.

[0107] 설명의 편의를 위해, 후술되는 실시예에서는, 코딩 트리 유닛 또는 코딩 유닛은 수직선 및 수평선 중 적어도 하나를 이용하여, 복수의 코딩 유닛으로 분할되는 것으로 가정한다.

[0108] 코딩 유닛의 파티셔닝이, 수직선(Vertical Line) 또는 수평선(Horizontal Line) 중 적어도 하나에 기초하여 수행된다고 가정할 때, 코딩 유닛을 파티셔닝하는 수직선 또는 수평선의 개수는 적어도 하나 이상일 수 있다. 일 예로, 하나의 수직선 또는 하나의 수평선을 이용하여, 코딩 트리 유닛 또는 코딩 유닛을 2개의 파티션으로 분할하거나, 두개의 수직선 또는 두개의 수평선을 이용하여, 코딩 유닛을 3개의 파티션으로 분할할 수 있다. 또는, 하나의 수직선 및 하나의 수평선을 이용하여, 코딩 유닛을 길이 및 너비가 1/2 인 4개의 파티션으로 분할할 수도 있다.

[0109] 코딩 트리 유닛 또는 코딩 유닛을 적어도 하나의 수직선 또는 적어도 하나의 수평선을 이용하여 복수의 파티션으로 분할하는 경우, 파티션들은 균일한 크기를 가질 수 있다. 또는, 어느 하나의 파티션이 나머지 파티션과 다른 크기를 갖거나, 각 파티션이 상이한 크기를 가질 수도 있다.

[0110] 후술되는 실시예들에서는, 코딩 유닛이 4개의 파티션으로 분할되는 것을, 쿼드 트리 기반의 분할이라 가정하고, 코딩 유닛이 2개의 파티션으로 분할되는 것을 바이너리 트리 기반의 분할이라 가정한다. 또한, 코딩 유닛이 3개의 파티션으로 분할되는 것을 트리플 트리 기반의 분할이라 가정한다. 또한 상기 적어도 2가지 이상의 분할 방식을 적용하여 분할되는 것을 멀티 트리 기반의 분할이라 가정한다.

[0111] 후술되는 도면에서는, 코딩 유닛을 분할하기 위해, 소정 개수의 수직선 또는 소정 개수의 수평선이 이용되는 것으로 도시할 것이나, 도시된 것보다 더 많은 수의 수직선 또는 더 많은 수의 수평선을 이용하여, 코딩 유닛을 도시된 것보다 더 많은 수의 파티션 또는 도시된 것보다 더 적은 수의 파티션으로 분할하는 것 역시 본 발명의 범주에 포함된다고 할 것이다.

[0112] 도 4는 발명이 적용되는 일 실시예로서, 쿼드 트리(Quad tree) 및 바이너리 트리(Binary tree) 분할(partitioning)이 허용되는 파티션 형태를 나타낸 도면이다.

[0113] 입력 영상 신호는 소정의 블록 단위로 복호화되며, 이와 같이 입력 영상 신호를 복호화하기 위한 기본 단위를 코딩 블록이라 한다. 코딩 블록은 인트라/인터 예측, 변환, 양자화를 수행하는 단위가 될 수 있다. 또한, 코딩 블록 단위로 예측 모드(예컨대, 화면 내 예측 모드 또는 화면 간 예측 모드)가 결정되고, 코딩 블록에 포함된 예측 블록들은, 결정된 예측 모드를 공유할 수 있다. 코딩 블록은 8x8 내지 64x64 범위에 속하는 임의의 크기를 가진 정방형 또는 비정방형 블록일 수 있고, 128x128, 256x256 또는 그 이상의 크기를 가진 정방형 또는 비정방형 블록일 수 있다.

[0114] 구체적으로, 코딩 블록은 쿼드 트리(quad tree)와 바이너리 트리(binary tree) 중 적어도 하나에 기초하여 계층적으로 분할될 수 있다. 여기서, 쿼드 트리 기반의 분할은 2Nx2N 코딩 블록이 4개의 NxN 코딩 블록으로 분할되는 방식(도 4(a))을, 바이너리 트리 기반의 분할은 하나의 코딩 블록이 2개의 코딩 블록으로 분할되는 방식을 각각 의미할 수 있다. 바이너리 트리 기반의 분할이 수행되었다 하더라도, 하위 뎁스에서는 정방형인 코딩 블록이 존재할 수 있다.

[0115] 바이너리 트리 기반의 분할은 대칭적으로 수행될 수도 있고, 비대칭적으로 수행될 수도 있다. 또한, 바이너리 트리 기반으로 분할된 코딩 블록은 정방형 블록일 수도 있고, 직사각형과 같은 비정방형 블록일 수도 있다. 일 예로, 바이너리 트리 기반의 분할이 허용되는 파티션 형태는 도 4 (b)에 도시된 예에서와 같이, 대칭형(symmetric)인 2NxN (수평 방향 비 정방 코딩 유닛) 또는 Nx2N (수직 방향 비정방 코딩 유닛)이 될 수 있다. 또한, 일 예로, 바이너리 트리 기반의 분할이 허용되는 파티션 형태는 도 4 (c)에 도시된 예에서와 같이, 비대칭형(asymmetric)인 nLx2N, nRx2N, 2NxnU 또는 2NxnD 중 적어도 하나를 포함할 수 있다.

[0116] 바이너리 트리 기반의 분할은, 대칭형 또는 비대칭 형태의 파티션 중 어느 하나만 제한적으로 허용될 수도 있다. 이 경우, 코딩 트리 유닛을, 정방형 블록으로 구성하는 것은 쿼드 트리 CU 파티셔닝에 해당하고, 코딩 트리 유닛을, 대칭형인 비정방형 블록으로 구성하는 것은 바이너리 트리 CU 파티셔닝에 해당할 수 있다. 코딩 트리 유닛을 정방형 블록과 대칭형 비정방형 블록으로 구성하는 것은 쿼드 및 바이너리 트리 CU 파티셔닝에 해당할 수 있다.

[0117] 이하, 상기 쿼드 트리 및 바이너리 트리에 기반한 분할 방식을 QTBT (Quad-Tree & Binary-Tree) 분할로 명명한다.

[0118] 쿼드 트리 및 바이너리 트리에 기반한 분할 결과, 더 이상 분할되지 않는 코딩 블록은 예측 블록 또는 변환 블록으로 이용될 수 있다. 즉, 쿼드 트리 및 바이너리 트리에 기반한 QTBT (Quad-Tree & Binary-Tree) 분할 방법에서는, 코딩 블록이 예측 블록이 되고, 예측 블록이 변환 블록이 될 수 있다. 일 예로, QTBT 분할 방법을 이용

한 경우, 코딩 블록 단위로 예측 영상을 생성하고, 코딩 블록 단위로 원본 영상과 예측 영상간의 차분인 잔차 신호가 변환될 수 있다. 여기서, 코딩 블록 단위로 예측 영상을 생성하는 것은, 코딩 블록을 기준으로 모션 정보가 결정되거나, 코딩 블록을 기준으로 하나의 인트라 예측 모드가 결정되는 것을 의미할 수 있다. 이에 따라, 코딩 블록은, 스킵 모드, 화면 내 예측 또는 화면 간 예측 중 적어도 하나를 이용하여 부호화될 수 있다.

[0119] 다른 예로, 코딩 블릭을 분할하여, 코딩 블록보다 작은 크기를 갖는 예측 블록 또는 변환 블록을 이용하는 것도 가능하다.

[0120] QTBT 분할 방법에서, BT는 대칭형 분할만이 허용되도록 설정될 수 있다. 다만, 블록 경계에서 오브젝트와 배경이 나누어지는 경우에도, 대칭형 이진 분할만을 허용한다면, 부호화 효율이 낮아질 수 있다. 이에 본 발명에서는, 부호화 효율을 높이기 위해, 코딩 블록을 비대칭으로 파티셔닝하는 방법을 다른 실시예로 후술하고자 한다. 비대칭 바이너리 트리 파티셔닝(Asymetric Binary Tree Partitioning)은 코딩 블록을 2개의 더 작은 코딩 블록으로 분할하는 것을 나타낸다. 비대칭 바이너리 트리 파티셔닝의 결과, 코딩 블록은 2개의 비대칭 형태의 코딩 블록으로 분할될 수 있다.

[0121] 바이너리 트리 기반의 분할은 쿼드 트리 기반의 분할이 더 이상 수행되지 않는 코딩 블록에 대해서 수행될 수 있다.바이너리 트리 기반으로 분할된 코딩 블록에 대해서는 쿼드 트리 기반의 분할이 더 이상 수행되지 않을 수 있다.

[0122] 또한, 하위 뎁스의 분할은 상위 뎁스의 분할 형태에 종속적으로 결정될 수 있다. 일 예로, 2개 이상의 뎁스에서 바이너리 트리 기반의 분할이 허용된 경우, 하위 뎁스에서는 상위 뎁스의 바이너리 트리 분할 형태와 동일한 형태의 바이너리 트리 기반의 분할만이 허용될 수 있다. 예컨대, 상위 뎁스에서 2NxN 형태로 바이너리 트리 기반의 분할이 수행된 경우, 하위 뎁스에서도 2NxN 형태의 바이너리 트리 기반의 분할이 수행될 수 있다. 또는, 상위 뎁스에서 Nx2N 형태로 바이너리 트리 기반의 분할이 수행된 경우, 하위 뎁스에서도 Nx2N 형태의 바이너리 트리 기반의 분할이 허용될 수 있다.

[0123] 반대로, 하위 뎁스에서, 상위 뎁스의 바이너리 트리 분할 형태와 상이한 형태의 바이너리 트리 기반의 분할만을 허용하는 것도 가능하다.

[0124] 시퀀스, 슬라이스, 코딩 트리 유닛 또는 코딩 유닛에 대해, 특정 형태의 바이너리 트리 기반의 분할만이 사용되도록 제한할 수도 있다. 일 예로, 코딩 트리 유닛에 대해 2NxN 또는 Nx2N 형태의 바이너리 트리 기반의 분할만이 허용되도록 제한할 수 있다. 허용되는 파티션 형태는 부호화기 또는 복호화기에 기 정의되어 있을 수도 있고, 허용되는 파티션 형태 또는 허용되지 않는 파티션 형태에 관한 정보를 부호화하여 비트스트림을 통해 시그널링할 수도 있다.

[0125] 도 5는 본 발명이 적용되는 일 실시예로서, 쿼드 트리 및 바이너리 트리 분할에 기반하여 코딩 블록을 계층적으로 분할하는 일예를 도시한 것이다.

[0126] 도 5에 도시된 바와 같이, 분할 깊이(split depth)가 k인 제1 코딩 블록 300은 쿼드 트리(quad tree)에 기반하여 복수의 제2 코딩 블록으로 분할될 수 있다. 예를 들어, 제2 코딩 블록 310 내지 340은 제1 코딩 블록의 너비와 높이의 절반 크기를 가진 정방형 블록이며, 제2 코딩 블록의 분할 깊이는 k+1로 증가될 수 있다.

[0127] 분할 깊이가 k+1인 제2 코딩 블록 310은 분할 깊이가 k+2인 복수의 제3 코딩 블록으로 분할될 수 있다. 제2 코딩 블록 310의 분할은 분할 방식에 따라 쿼트 트리 또는 바이너리 트리 중 어느 하나를 선택적으로 이용하여 수행될 수 있다. 여기서, 분할 방식은 쿼드 트리 기반으로의 분할을 지시하는 정보 또는 바이너리 트리 기반의 분할을 지시하는 정보 중 적어도 하나에 기초하여 결정될 수 있다.

[0128] 제2 코딩 블록 310이 쿼트 트리 기반으로 분할되는 경우, 제2 코딩 블록 310은 제2 코딩 블록의 너비와 높이의 절반 크기를 가진 4개의 제3 코딩 블록 310a으로 분할되며, 제3 코딩 블록 310a의 분할 깊이는 k+2로 증가될 수 있다. 반면, 제2 코딩 블록 310이 바이너리 트리 기반으로 분할되는 경우, 제2 코딩 블록 310은 2개의 제3 코딩 블록으로 분할될 수 있다. 이때, 2개의 제3 코딩 블록 각각은 제2 코딩 블록의 너비와 높이 중 어느 하나가 절반 크기인 비정방형 블록이며, 분할 깊이는 k+2로 증가될 수 있다. 제2 코딩 블록은 분할 방향에 따라 가로 방향 또는 세로 방향의 비정방형 블록으로 결정될 수 있고, 분할 방향은 바이너리 트리 기반의 분할이 세로 방향인지 또는 가로 방향인지에 관한 정보에 기초하여 결정될 수 있다.

[0129] 한편, 제2 코딩 블록 310은 쿼드 트리 또는 바이너리 트리에 기반하여 더 이상 분할되지 않는 말단 코딩 블록으로 결정될 수도 있고, 이 경우 해당 코딩 블록은 예측 블록 또는 변환 블록으로 이용될 수 있다.

[0130] 제3 코딩 블록 310a은 제2 코딩 블록 310의 분할과 마찬가지로 말단 코딩 블록으로 결정되거나, 쿼드 트리 또는 바이너리 트리에 기반하여 추가적으로 분할될 수 있다.

[0131] 한편, 바이너리 트리 기반으로 분할된 제3 코딩 블록 310b은 추가적으로 바이너리 트리에 기반하여 세로 방향의 코딩 블록(310b-2) 또는 가로 방향의 코딩 블록(310b-3)으로 더 분할될 수도 있고, 해당 코딩 블록의 분할 깊이는 k+3으로 증가될 수 있다. 또는, 제3 코딩 블록 310b는 바이너리 트리에 기반하여 더 이상 분할되지 않는 말단 코딩 블록(310b-1)으로 결정될 수 있고, 이 경우 해당 코딩 블록(310b-1)은 예측 블록 또는 변환 블록으로 이용될 수 있다. 다만, 상술한 분할 과정은 쿼드 트리 기반의 분할이 허용되는 코딩 블록의 크기/깊이에 관한 정보, 바이너리 트리 기반의 분할이 허용되는 코딩 블록의 크기/깊이에 대한 정보 또는 바이너리 트리 기반의 분할이 허용되지 않는 코딩 블록의 크기/깊이에 대한 정보 중 적어도 하나에 기초하여 제한적으로 수행될 수 있다.

[0132] 코딩 블록이 가질 수 있는 크기는 소정 개수로 제한되거나, 소정 단위 내 코딩 블록의 크기는 고정된 값을 가질 수도 있다. 일 예로, 시퀀스 내 코딩 블록의 크기 또는 픽처 내 코딩 블록의 크기는, 256x256, 128x128 또는 32x32로 제한될 수 있다. 시퀀스 또는 픽처 내 코딩 블록의 크기를 나타내는 정보가 시퀀스 헤더 또는 픽처 헤더를 통해 시그널링 될 수 있다.

[0133] 쿼드 트리 및 바이너리 트리에 기반한 분할 결과, 코딩 유닛은, 정방형 또는 임의 크기의 직사각형을 띨 수 있다.

[0134] 도 6은 본 발명이 적용되는 일 실시예로서, 쿼드 트리 및 대칭형 바이너리 트리 분할에 기반하여 코딩 블록을 계층적으로 분할하는 일예를 도시한 것이다.

[0135] 도 6은 특정 형태, 예를 들어 대칭형 바이너리 트리 기반의 분할만이 허용된 예를 나타낸 도면이다. 도 6의 (a)는 Nx2N 형태의 바이너리 트리 기반의 분할만이 허용되도록 제한된 예를 나타낸다. 예를 들어, 뎁스 1 코딩 블록 601은 뎁스 2에서 2개의 Nx2N 블록 (601a, 601b)으로 분할되고, 또한, 뎁스 2 코딩 블록 602는 뎁스 3에서 2개의 Nx2N 블록 (602a, 602b)로 분할 가능하다.

[0136] 도 6의 (b)는 2NxN 형태의 바이너리 트리 기반의 분할만이 허용되도록 제한된 예를 나타낸다. 예를 들어, 뎁스 1 코딩 블록 603은 뎁스 2에서 2개의 2NxN 블록 (603a, 603b)으로 분할되고, 또한, 뎁스 2 코딩 블록 604는 뎁스 3에서 2개의 2NxN 블록 (604a, 604b)로 분할 가능하다.

[0137] 도 6의 (c)는 대칭형 바이너리 트리로 분할된 블록을 다시 대칭형 바이너리 트리로 분할하는 예를 나타낸다. 예를 들어, 뎁스 1 코딩 블록 605는, 뎁스 2에서 2개의 Nx2N 블록 (605a, 605b)으로 분할되고, 또한, 상기 분할후 생성된 뎁스 2 코딩 블록 605a는 뎁스 3에서 2개의 Nx2N 블록 (605a1, 605a2)로 분할 가능하다. 상기 분할 방식은 대칭형 바이너리 트리 분할에 의해 생성된 2NxN 코딩 블록에 대해서도 동일하게 적용 가능하다.

[0138] 상기 쿼드 트리 또는 바이너리 트리 기반의 적응적 분할을 구현하기 위해 쿼드 트리 기반의 분할을 지시하는 정보, 쿼드 트리 기반의 분할이 허용되는 코딩 블록의 크기/깊이에 관한 정보, 바이너리 트리 기반의 분할을 지시하는 정보, 바이너리 트리 기반의 분할이 허용되는 코딩 블록의 크기/깊이에 대한 정보, 바이너리 트리 기반의 분할이 허용되지 않는 코딩 블록의 크기/깊이에 대한 정보 또는 바이너리 트리 기반의 분할이 세로 방향인지 또는 가로 방향인지에 관한 정보 등이 이용될 수 있다. 일 예로, quad_split_flag는 코딩 블록이 4개의 코딩 블록으로 분할되는지 여부를 나타내고, binary_split_flag는 코딩 블록이 2개의 코딩 블록으로 분할되는지 여부를 나타낼 수 있다. 코딩 블록이 2개의 코딩 블록으로 분할되는 경우, 코딩 블록의 분할 방향이 수직 방향인지 또는 수평 방향인지 여부를 나타내는 is_hor_split_flag가 시그널링될 수 있다.

[0139] 또한, 코딩 트리 유닛 또는 소정의 코딩 유닛에 대해, 바이너리 트리 분할이 허용되는 횟수, 바이너리 트리 분할이 허용되는 깊이 또는 바이너리 트리 분할이 허용된 뎁스의 개수 등이 획득될 수 있다. 상기 정보는 코딩 트리 유닛 또는 코딩 유닛 단위로 부호화되어, 비트스트림을 통해 복호화기로 전송될 수 있다.

[0140] 일 예로, 비트스트림을 통해, 바이너리 트리 분할이 허용되는 최대 뎁스를 나타내는 신택스 'max_binary_depth_idx_minus1'가 비트스트림을 통해 부호화/복호화될 수 있다. 이 경우, max_binary_depth_idx_minus1+1이 바이너리 트리 분할이 허용되는 최대 뎁스를 가리킬 수 있다.

[0141] 또한, 전술한 도 6 (c) 예를 살펴보면, 뎁스 2인 코딩 유닛 (예, 605a, 605b) 및 뎁스 3인 코딩 유닛 (예, 605a1, 605a2)에 대해 바이너리 트리 분할이 수행된 결과가 도시되었다. 이에 따라, 코딩 트리 유닛 내 바이너리 트리 분할이 수행된 횟수(예, 2회)를 나타내는 정보, 코딩 트리 유닛 내 바이너리 트리 분할이 허용된 최대

뎁스(예, 뎁스 3)를 나타내는 정보 또는 코딩 트리 유닛 내 바이너리 트리 분할이 허용된 뎁스의 개수(예, 2개, 뎁스 2 및 뎁스 3)를 나타내는 정보 중 적어도 하나가 비트스트림을 통해 부호화/복호화될 수 있다.

[0142] 다른 예로, 바이너리 트리 분할이 허용되는 횟수, 바이너리 트리 분할이 허용되는 깊이 또는 바이너리 트리 분할이 허용된 뎁스의 개수 중 적어도 하나는 시퀀스, 슬라이스별로 획득될 수 있다. 일 예로, 상기 정보는, 시퀀스, 픽처 또는 슬라이스 단위로 부호화되어 비트스트림을 통해 전송될 수 있다. 이에 따라, 제1 슬라이스 및 제2 슬라이스의, 바이너리 트리 분할 횟수, 바이너리 트리 분할이 허용되는 최대 뎁스 또는 바이너리 트리 분할이 허용되는 뎁스의 개수 중 적어도 하나가 상이할 수 있다. 일 예로, 제1 슬라이스에서는, 하나의 뎁스에서만 바이너리 트리 분할이 허용되는 반면, 제2 슬라이스에서는, 두개의 뎁스에서 바이너리 트리 분할이 허용될 수 있다.

[0143] 또 다른 일 예로, 슬라이스 또는 픽처의 시간레벨 식별자(Temporal_ID)에 따라 바이너리 트리 분할이 허용되는 횟수, 바이너리 트리 분할이 허용되는 깊이 또는 바이너리 트리 분할이 허용되는 뎁스의 개수 중 적어도 하나를 상이하게 설정할 수도 있다. 여기서, 시간레벨 식별자(Temporal_ID)는, 시점(view), 공간(spatial), 시간(temporal) 또는 화질(quality) 중 적어도 하나 이상의 스케일러빌리티(Scalability)를 갖는 영상의 복수개의 레이어 각각을 식별하기 위한 것이다.

[0144] 또한, 바이너리 파티셔닝으로 파티션된 CU에서는 Transform skip을 사용하지 않도록 제한할 수도 있다. 또는 비정방형으로 파티션된 CU에서는 수평 방향 또는 수직 방향 중 적어도 어느 하나의 방향에서만 transformskip을 적용할 수도 있다. 수평방향 transform skip만 적용하는 것은, 수평 방향으로 transform 수행없이 스케일링과 양자화만 수행하고, 수직 방향으로 DCT 나 DST 등 적어도 어느 하나의 transform을 특정하여 변환을 수행하는 것을 나타낸다.

[0145] 이와 마찬가지로, 수직방향 transform skip만 적용하는 것은 수평 방향으로 DCT 나 DST 등 적어도 어느 하나의 transform을 특정하여 변환을 수행하고, 수직 방향으로 transform 수행없이 스케일링과 양자화만 수행하는 것을 나타낸다. 수평 방향 transform skip을 적용할지를 알려주는 신택스 hor_transform_skip_flag과 수직 방향 transform skip을 적용 여부를 알려주는 신택스 ver_transform_skip_flag을 시그날링할 수도 있다.

[0146] 수평방향 또는 수직방향 중 적어도 어느 하나에 transform skip을 적용할 때, CU의 형태에 따라 어느 방향으로 transform skip을 적용할 지를 시그날링 할 수도 있다.구체적으로 예를 들어, 2NxN 형태의 CU인 경우에 수평 방향으로 transform을 수행하고 수직 방향으로 transform skip을 적용할 수도 있으며, Nx2N 형태의 CU인 경우에 수평 방향으로 transform skip을 적용하고 수직 방향을 transform을 수행할 수도 있다. 여기서 transform은 DCT 또는 DST 중 적어도 어느 하나일 수 있다.

[0147] 또 다른 예를 들어, 2NxN 형태의 CU인 경우에 수직 방향으로 transform을 수행하고 수평 방향으로 transform skip을 적용할 수도 있으며, Nx2N 형태의 CU인 경우에 수직 방향으로 transform skip을 적용하고 수평 방향을 transform을 수행할 수도 있다. 여기서 transform은 DCT 또는 DST 중 적어도 어느 하나일 수 있다.

[0148] 도 7은 본 발명이 적용되는 일 실시예로서, 비대칭형 바이너리 트리 분할이 허용되는 파티션 형태를 나타낸 도면이다.2Nx2N 코딩 블록은 너비 비가 n:(1-n)인 2개의 코딩 블록 또는 높이 비가 n:(1-n)인 2개의 코딩 블록으로 분할될 수 있다. 여기서, n은 0보다 크고 1보다 작은 실수를 나타낼 수 있다.

[0149] 도 7에서는, 예를 들어 코딩 블록에 비대칭 바이너리 트리 파티셔닝이 적용됨에 따라, 너비 비가 1:3 인 2개의 코딩 블록 (701, 702), 또는 3:1인 2개의 코딩 블록 (703, 704), 또는 높이 비가 1:3 인 2개의 코딩 블록 (705, 706) 또는 3:1인 2개의 코딩 블록(707, 708)이 생성되는 것으로 도시되었다.

[0150] 구체적으로, WxH 크기의 코딩 블록이 수직 방향으로 분할됨에 따라, 너비가 1/4W인 좌측 파티션 및 너비가 3/4W인 우측 파티션이 생성될 수 있다. 위와 같이, 좌측 파티션의 너비가 우측 파티션의 너비보다 작은 분할 형태를 nLx2N 바이너리 파티션이라 호칭할 수 있다.

[0151] WxH 크기의 코딩 블록이 수직 방향으로 분할됨에 따라, 너비가 3/4W인 좌측 파티션 및 너비가 1/4W인 우측 파티션이 생성될 수도 있다. 위와 같이, 우측 파티션의 너비가 좌측 파티션의 너비보다 작은 분할 형태를 nRx2N 바이너리 파티션이라 호칭할 수 있다.

[0152] WxH 크기의 코딩 블록이 수평 방향으로 분할됨에 따라, 높이가 1/4H인 상단 파티션 및 높이가 3/4H인 하단 파티션이 생성될 수 있다. 위와 같이, 상단 파티션의 높이가 하단 파티션의 높이보다 작은 분할 형태를 2NxnU 바이너리 파티션이라 호칭할 수 있다.

[0153] WxH 크기의 코딩 블록이 수평 방향으로 분할됨에 따라, 높이가 3/4H인 상단 파티션 및 높이가 1/4H인 하단 파티션이 생성될 수 있다. 위와 같이, 하단 파티션의 높이가 상단 파티션의 높이보다 작은 분할 형태를 2NxnD 바이너리 파티션이라 호칭할 수 있다.

[0154] 도 7에서는 두 코딩 블록간의 너비 비 또는 높이 비가 1:3 또는 3:1인 경우를 예시하였으나, 비대칭 바이너리 트리 파티셔닝에 의해 생성되는 두 코딩 블록 간 너비 비 또는 높이 비가 이에 한정되는 것은 아니다. 코딩 블록은 도 7에 도시된 것과 상이한 너비 비 또는 상이한 높이 비를 갖는 2개의 코딩 블록으로 분할될 수도 있다.

[0155] 비대칭 바이너리 트리 파티셔닝을 이용하는 경우, 코딩 블록의 비대칭 바이너리 파티션 형태는 비트스트림을 통해 시그널링되는 정보에 기초하여 결정될 수 있다.일 예로, 코딩 블록의 분할 형태는 코딩 블록의 분할 방향을 나타내는 정보 및 코딩 블록이 분할됨에 따라 생성되는 제1 파티션이 제2 파티션보다 작은 크기를 갖는지 여부를 나타내는 정보를 기초로 결정될 수 있다.

[0156] 코딩 블록의 분할 방향을 나타내는 정보는, 코딩 블록이 수직 방향으로 분할되었는지 또는 수평 방향으로 분할되었는지 여부를 나타내는 1비트의 플래그일 수 있다. 일 예로, hor_binary_flag 는 코딩 블록이 수평 방향으로 분할되었는지 여부를 나타낼 수 있다. hor_binary_flag의 값이 1인 것은, 코딩 블록이 수평 방향으로 분할됨을 나타내고, hor_binary_flag의 값이 0인 것은, 코딩 블록이 수직 방향으로 분할됨을 나타낼 수 있다. 또는 코딩 블록이 수직 방향으로 분할되었는지 여부를 나타내는 ver_binary_flag가 이용될 수도 있다.

[0157] 제1 파티션이 제2 파티션보다 작은 크기를 갖는지 여부를 나타내는 정보는, 1비트의 플래그일 수 있다. 일 예로, is_left_above_small_part_flag는 코딩 블록이 분할됨에 따라 생성된 좌측 또는 상단 파티션의 크기가 우측 또는 하측 파티션 보다 작은지 여부를 나타낼 수 있다. is_left_above_small_part_flag의 값이 1인 것은 좌측 또는 상단 파티션의 크기가 우측 또는 하단 파티션보다 작은 것을 의미하고, is_left_above_small_part_flag의 값이 0인 것은 좌측 또는 상단 파티션의 크기가 우측 또는 하단 파티션보다 큰 것을 의미할 수 있다. 또는, 우측 또는 하단 파티션의 크기가 좌측 또는 상단 파티션보다 작은지 여부를 나타내는 is_right_bottom_small_part_flag를 사용할 수도 있다.

[0158] 또는, 제1 파티션 및 제2 파티션 간의 너비비, 높이비 또는 넓이비를 나타내는 정보를 사용하여 제1 파티션 및 제2 파티션의 크기를 결정할 수도 있다.

[0159] hor_binary_flag의 값이 0이고, is_left_above_small_part_flag의 값이 1인 것은, nLx2N 바이너리 파티션을 나타내고, hor_binary_flag의 값이 0이고, is_left_above_small_part_flag의 값이 0인 것은, nRx2N 바이너리 파티션을 나타낼 수 있다. 또한, hor_binary_flag의 값이 1이고, is_left_above_small_part_flag의 값이 1인 것은, 2NxnU 바이너리 파티션을 나타내고, hor_binary_flag의 값이 1이고, is_left_above_small_part_flag의 값이 0인 것은 2NxnD 바이너리 파티션을 나타낼 수 있다.

[0160] 다른 예로, 코딩 블록의 비대칭 바이너리 파티션 형태는, 코딩 블록의 파티션 형태를 지시하는 인덱스 정보에 의해 결정될 수도 있다. 여기서, 인덱스 정보는 비트스트림을 통해 시그널링되는 정보로, 고정된 길이(즉, 고정된 비트 수)로 부호화될 수도 있고, 가변 길이로 부호화될 수도 있다. 일 예로, 하기 표 1은 비대칭 바이너리 파티션별 파티션 인덱스를 나타낸 것이다.

표 1

[0161]

	Asymetric partition index	Binarization
nLx2N	0	0
nRx2N	1	10
2NxnU	2	100
2NxnD	3	111

[0162] 비대칭 바이너리 트리 파티셔닝은 QTBT 분할 방법에 종속적으로 이용될 수 있다.일 예로, 코딩 블록에 더 이상 쿼드 트리 분할 또는 바이너리 트리 분할이 적용되지 않는 경우, 해당 코딩 블록에 비대칭 바이너리 트리 분할을 적용할 것인지 여부가 결정될 수 있다. 여기서, 코딩 블록에 비대칭 바이너리 트리 분할을 적용할 것인지 여부는 비트스트림을 통해 시그널링되는 정보에 의해 결정될 수 있다. 예컨대, 상기 정보는 1비트의 플래그 'asymmetric_binary_tree_flag'일 수 있고, 상기 플래그에 기초하여, 코딩 블록에 비대칭 바이너리 트리 분할이 적용되는지 여부가 결정될 수 있다.또는, 코딩 블록이 2개의 블록으로 분할되는 것으로 결정되는 경우, 그 분할 형태가 바이너리 트리 분할인지 또는 비대칭 바이너리 트리 분할인지 여부가 결정될 수도 있다. 여기서, 코딩

블록의 분할 형태가 바이너리 트리 분할인지 또는 비대칭 바이너리 트리 분할인지 여부는 비트스트림을 통해 시그널링되는 정보에 의해 결정될 수 있다. 예컨대, 상기 정보는 1비트의 플래그 'is_asymmetric_split_flag'일 수 있고, 상기 플래그에 기초하여, 코딩 블록이 대칭 또는 비대칭 형태로 분할되는지 여부가 결정될 수 있다.

[0163] 다른 예로, 대칭형 바이너리 파티션 및 비대칭형 바이너리 파티션에 서로 다른 인덱스를 할당하고, 인덱스 정보에 따라, 코딩 블록이 대칭 형태 또는 비대칭 형태로 분할되는지 여부를 결정할 수도 있다. 일 예로, 표 2는 대칭형 바이너리 파티션 및 비대칭형 바이너리 파티션에 각기 다른 인덱스가 할당된 예를 나타낸 것이다.

표 2

[0164]

	Binary partition index	Binarization
2NxN (수평 방향 바이너리 파티션)	0	0
Nx2N(수직 방향 바이너리 파티션)	1	10
nLx2N	2	110
nRx2N	3	1110
2NxnU	4	11110
2NxnD	5	11111

[0165] 코딩 트리 블록 또는 코딩 블록은, 쿼드 트리 분할, 바이너리 트리 분할 또는 비대칭 바이너리 트리 분할을 통해 복수의 코딩 블록으로 세분화될 수 있다. 일 예로, 도 8은 QTBT 및 비대칭 바이너리 트리 분할을 이용하여 코딩 블록이 복수의 코딩 블록으로 분할되는 예를 나타낸 도면이다. 도 9를 참조하면, 첫번째 그림의 뎁스 2 파티셔닝, 두번째 그림의 뎁스 3 파티셔닝, 세번째 그림의 뎁스 3 파티셔닝에서 각각 비대칭 바이너리 트리 분할이 수행된 것을 확인할 수 있다.비대칭 바이너리 트리 파티셔닝을 통해 분할된 코딩 블록은 더 이상 분할되지 않도록 제한될 수 있다. 일 예로, 비대칭 바이너리 트리 파티셔닝을 통해 생성된 코딩 블록에는 쿼드 트리, 바이너리 트리 또는 비대칭 바이너리 트리 관련 정보가 부호화/복호화되지 않을 수 있다. 즉, 비대칭 바이너리 트리 파티셔닝을 통해 생성된 코딩 블록에 대해서는, 쿼드 트리 분할 여부를 나타내는 플래그, 바이너리 트리 분할 여부를 타나내는 플래그, 비대칭 바이너리 트리 분할 여부를 나타내는 플래그, 바이너리 트리 또는 비대칭 바이너리 트리 분할 방향을 나타내는 플래그, 또는 비대칭 바이너리 파티션을 나타내는 인덱스 정보 등의 신택스의 부호화/복호화가 생략될 수 있다.

[0166] 다른 예로, 바이너리 트리 파티셔닝을 허용할 것인지 여부는 QTBT의 허용 여부에 종속적으로 결정될 수 있다. 일 예로, QTBT에 기초한 분할 방법이 사용되지 않는 픽쳐 또는 슬라이스에서는 비대칭 바이너리 트리 파티셔닝이 사용되지 않도록 제한될 수 있다.

[0167] 비대칭 바이너리 트리 파티셔닝이 허용되는지 여부를 나타내는 정보가 블록 단위, 슬라이스 단위 또는 픽처 단위로 부호화되어 시그널링될 수도 있다. 여기서, 비대칭 바이너리 트리 파티셔닝이 허용되는지 여부를 나타내는 정보는 1비트의 플래그일 수 있다. 일 예로, is_used_asymmetric_QTBT_enabled_flag의 값이 0인 것은, 비대칭 바이너리 트리 파티셔닝이 사용되지 않음을 나타낼 수 있다. 픽처 단위 또는 슬라이스 단위로 바이너리 트리 파티셔닝이 사용되지 않는 경우, is_used_asymmetric_QTBT_enabled_flag를 시그널링하지 않고, 그 값을 0으로 설정할 수도 있다.

[0168] 도 8은 본 발명이 적용되는 일 실시예로서, 쿼드 트리 및 대칭형/비대칭형 바이너리 트리 분할에 기반한 코딩 블록의 분할 형태를 예시한 것이다.

[0169] 도 8의 (a)는 nLx2N 형태의 비대칭형 바이너리 트리 기반의 분할이 허용된 예를 나타낸다. 예를 들어, 뎁스 1 코딩 블록 801은 뎁스 2에서 비대칭형 2개의 nLx2N 블록 (801a, 801b)으로 분할되고, 또한, 뎁스 2 코딩 블록 801b는 뎁스 3에서 대칭형 2개의 Nx2N 블록 (801b1, 801b2)로 분할된 예를 도시한 것이다.

[0170] 도 8의 (b)는, nRx2N 형태의 비대칭형 바이너리 트리 기반의 분할이 허용된 예를 나타낸다. 예를 들어, 뎁스 2 코딩 블록 802는 뎁스 3에서 비대칭형 2개의 nRx2N 블록 (802a, 802b)으로 분할된 예를 도시한 것이다.

[0171] 도 8의 (c)는 2NxnU 형태의 비대칭형 바이너리 트리 기반의 분할이 허용된 예를 나타낸다. 예를 들어, 뎁스 2 코딩 블록 803은 뎁스 3에서 비대칭형 2개의 2NxnU 블록 (803a, 803b)으로 분할된 예를 도시한 것이다.

[0172] 코딩 블록의 크기, 형태, 분할 깊이 또는 분할 형태 등에 기초하여, 코딩 블록에 허용되는 분할 형태가 결정될 수도 있다. 일 예로, 쿼드 트리 분할에 의해 생성된 코딩 블록 및 바이너리 트리 분할에 의해 생성된 코딩 블록

사이 허용되는 분할 타입, 파티션 형태 또는 파티션 개수 중 적어도 하나는 상이할 수 있다.

[0173] 일 예로, 코딩 블록이 쿼드 트리 분할에 의해 생성된 것일 경우, 해당 코딩 블록에는, 쿼드 트리 분할, 바이너리 트리 분할 및 비대칭 바이너리 트리 분할 모두 허용될 수 있다. 즉, 코딩 블록이 쿼드 트리 분할에 기초하여 생성된 것일 경우, 코딩 블록에는 도 10에 나타난 모든 파티션 형태가 적용될 수 있다.일 예로, 2Nx2N 파티션은 코딩 블록이 더 이상 분할되지 않는 경우를 나타내고, NxN은 코딩 블록이 쿼드트리 분할되는 경우를 나타내며, Nx2N 및 2NxN은 코딩 블록이 바이너리 트리 분할되는 경우를 나타낼 수 있다. 또한, nLx2N, nRx2N, 2NxnU 및 2NxnD는 코딩 블록이 비대칭 바이너리 트리 분할되는 경우를 나타낼 수 있다.

[0174] 반면, 코딩 블록이 바이너리 트리 분할에 의해 생성된 것일 경우, 해당 코딩 블록에는 비대칭 바이너리 트리 분할을 제한할 수 있다. 즉, 코딩 블록이 바이너리 트리 분할에 기초하여 생성된 것일 경우, 코딩 블록에는 도 7에 도시된 파티션 형태들 중 비대칭 파티션 형태(nLx2N, nRx2N, 2NxnU, 2NxnD)을 적용하는 것이 제한될 수 있다.

[0175] 도 9는 본 발명이 적용되는 일 실시예로서, 쿼드 트리 및 바이너리 트리 분할에 기반한 코딩 블록 분할 방법에 대한 흐름도이다.

[0176] 뎁스 k 코딩 블록을, 뎁스 k+1 코딩 블록으로 분할하는 것으로 가정한다. 우선, 뎁스 k 현재 블록에 쿼드 트리 분할이 적용되는 지를 판단한다(S910). 만약 쿼드 트리 분할이 적용되었다면, 현재 블록을 4개의 정방형 블록으로 분할한다(S920). 반면, 만약 쿼드 트리 분할이 적용되지 않았다면, 현재 블록에 바이너리 트리 분할이 적용되는 지를 판단한다(S930). 만약 바이너리 트리 분할도 적용되지 않았다면, 현재 블록은 분할 없이 뎁스 k+1 코딩 블록이 된다. 상기 S930 판단 결과, 현재 블록에 바이너리 트리 분할이 적용되었다면, 대칭형 바이너리 분할 또는 비대칭형 바이너리 분할 중 어느 방식이 적용되는 지를 확인한다(S940). 상기 S940 판단 결과에 따라, 현재 블록에 적용되는 파티션 형태를 결정한다(S950). 예를 들어, 상기 S950 단계에 적용되는 파티션 형태는, 대칭형인 경우 도 4(b) 형태중 어느 하나, 또는 비대칭형인 경우 도4(c) 형태 중 어느 하나가 될 수 있다. 상기 S950을 통해, 결정된 파티션 형태에 따라, 현재 블록을 2개의 뎁스 k+1 코딩 블록으로 분할하게 된다(S960).

[0177] 도 10은 본 발명이 적용되는 일 실시예로서, 쿼드 트리 및 바이너리 트리 분할이 적용되는 네트워크 추상화 계층 (NAL)에 포함되는 신택스 요소(syntax element)를 예를 들어 도시한 것이다.

[0178] 본 발명이 적용되는 압축된 영상은, 예를 들어 네트워크 추상화 계층 (Network Abstract Layer, 이하 'NAL' 이라 함) 단위로 패킷화 되어 전송 매체를 통해 전송될 수 있다. 단, 본 발명은 NAL에 한정되지 않으며, 향후 개발될 다양한 데이터 전송 방식에도 적용 가능하다. 본 발명이 적용되는 NAL 유닛은, 예를 들어, 도 10에 도시된 바와 같이 비디오 파라미터 셋 (VPS), 시퀀스 파라미터 셋(SPS), 픽쳐 파라미터 셋(PPS) 및 적어도 하나 이상의 슬라이스 셋(Slice)을 포함할 수 있다.

[0179] 예를 들어, 도 10에서는 시퀀스 파라미터 셋(SPS)에 포함된 신택스 요소를 도시하였으나, 픽쳐 파라미터 셋(PPS) 또는 슬라이스 셋(Slice)에 신택스 요소를 포함하는 것도 가능하다. 또한, 신택스 요소별로 시퀀스 단위 또는 픽쳐 단위에 공통적으로 적용될 신택스 요소는 시퀀스 파라미터 셋(SPS) 또는 픽쳐 파라미터 셋(PPS)에 포함되도록 할 수 있다. 반면, 해당 슬라이스에만 적용되는 신택스 요소는 슬라이스 셋(Slice)에 포함되는 것이 바람직하다. 따라서, 이는 부호화 성능 및 효율을 고려하여 선택이 가능하다.

[0180] 관련하여, 쿼드 트리 및 바이너리 트리 분할이 적용되는 신택스 요소를 설명하면 다음과 같다. 도 10에 도시된 모든 신택스 요소를 필수 요소로 설정하는 것도 가능하지만, 부호화 효율 및 성능을 고려하여, 이중 신택스 요소를 선택적으로 설정하는 것도 가능하다.

[0181] 일 예로, 'quad_split_flag'는 코딩 블록이 4개의 코딩 블록으로 분할되는지 여부를 나타낸다. 'binary_split_flag'는 코딩 블록이 2개의 코딩 블록으로 분할되는지 여부를 나타낼 수 있다. 코딩 블록이 2개의 코딩 블록으로 분할되는 경우, 코딩 블록의 분할 방향이 수직 방향인지 또는 수평 방향인지 여부를 나타내는 'is_hor_split_flag'가 시그널링될 수 있다. “is_hor_split_flag = 1” 이면 수평방향을 “is_hor_split_flag = 0” 이면 수직방향을 나타내는 것으로 정의할 수 있다.

[0182] 또한, 다른 대안으로, 'isUseBinaryTreeFlag'를 통해 현재 블록에 바이너리 트리 파티셔닝이 적용 여부를 나타내고, 또한, 코딩 블록의 분할 방향을 나타내는 신택스 요소로서, 'hor_binary_flag' 는 코딩 블록이 수평 방향으로 분할되었는지 여부를 나타낼 수 있다. 예를 들어, “hor_binary_flag = 1”인 경우, 코딩 블록이 수평 방향으로 분할됨을 나타내고, “hor_binary_flag = 0”인 경우, 코딩 블록이 수직 방향으로 분할됨을 나타낼 수 있다. 또는 'hor_binary_flag' 대신 코딩 블록이 수직 방향으로 분할되었는지 여부를 나타내는 ver_binary_flag

가 이용하여 동일한 방식으로 설정할 수 있다.

[0183] 또한, 바이너리 트리 분할이 허용되는 최대 뎁스를 나타내는 신택스 요소로서, 'max_binary_depth_idx_minus1'을 정의할 수 있다. 예를 들어, “max_binary_depth_idx_minus1 + 1”이 바이너리 트리 분할이 허용되는 최대 뎁스를 가리킬 수 있다.

[0184] 또한, 수평 방향 transform skip을 적용할지를 알려주는 신택스 요소로서, 'hor_transform_skip_flag'과 수직 방향 transform skip을 적용 여부를 알려주는 신택스 요소로 'ver_transform_skip_flag'을 설정할 수도 있다.

[0185] 또한, 비대칭 바이너리 트리 파티셔닝이 허용되는지 여부를 나타내는 신택스 요소로서, 'is_used_asymmetric_QTBT_enabled_flag'을 정의할 수 있다. 예를 들어, “is_used_asymmetric_QTBT_enabled_flag = 1” 이면, 비대칭 바이너리 트리 파티셔닝이 사용되었음을 나타내고, “is_used_asymmetric_QTBT_enabled_flag = 0” 이면, 비대칭 바이너리 트리 파티셔닝이 사용되지 않았음을 나타낼 수 있다. 반면, 픽처 단위 또는 슬라이스 단위로 바이너리 트리 파티셔닝이 사용되지 않는 경우, is_used_asymmetric_QTBT_enabled_flag를 시그널링하지 않고, 그 값을 0으로 설정할 수도 있다. 또한, 다른 대안으로, 'asymmetric_binary_tree_flag'를 통해 현재 블록에 비대칭형 바이너리 트리 파티셔닝이 적용되는 지를 나타낼 수 있다.

[0186] 또한, 비대칭형 바이너리 트리 분할을 나타내는 신택스 요소로서, 'is_left_above_small_part_flag'는 코딩 블록이 분할됨에 따라 생성된 좌측 또는 상단 파티션의 크기가 우측 또는 하측 파티션 보다 작은지 여부를 나타낼 수 있다. 예를 들어, “is_left_above_small_part_flag =1” 인 경우, 좌측 또는 상단 파티션의 크기가 우측 또는 하단 파티션보다 작은 것을 의미하고, “is_left_above_small_part_flag = 0”인 경우, 좌측 또는 상단 파티션의 크기가 우측 또는 하단 파티션보다 큰 것을 의미할 수 있다. 또는, 'is_left_above_small_part_flag' 대신, 우측 또는 하단 파티션의 크기가 좌측 또는 상단 파티션보다 작은지 여부를 나타내는 'is_right_bottom_small_part_flag'를 사용할 수도 있다.

[0187] 관련하여, 상기 신택스 요소들을 조합하여 코딩 블록의 비대칭형 바이너리 파티션 형태를 정의하는 것이 가능하다. 예를 들어, “hor_binary_flag = 0” 이고 “is_left_above_small_part_flag =1” 이면 nLx2N 바이너리 파티션을 나타내고, “hor_binary_flag = 0”이고, “is_left_above_small_part_flag = 0” 이면 nRx2N 바이너리 파티션을 나타내는 것으로 설정할 수 있다. 또한, “hor_binary_flag = 1”이고 “is_left_above_small_part_flag = 1” 이면 2NxnU 바이너리 파티션을 나타내고, “hor_binary_flag = 1”이고 “is_left_above_small_part_flag = 0”이면 2NxnD 바이너리 파티션을 나타낼 수 있다. 마찬가지로, 상기 'ver_binary_flag' 및 'is_right_bottom_small_part_flag'의 조합을 이용하여 비대칭형 바이너리 파티션 형태를 나타낼 수 도 있다.

[0188] 또한, 다른 대안으로, Asymetric_partition_index'에 의해 전술한 표 1의 인덱스를 표시하거나, 또는 'Binary_partition_index'에 의해 전술한 표 2의 인덱스를 표시함에 의해, 코딩 블록의 비대칭형 바이너리 파티션 형태를 정의하는 것이 가능하다.

[0189] 상술한 예에서 살펴본 바와 같이, 코딩 유닛(또는 코딩 트리 유닛)은 적어도 하나의 수직선 또는 수평선 등에 의해 재귀적으로 분할될 수 있다. 일 예로, 쿼드 트리 분할은, 수평선 및 수직선을 이용하여 코딩 블록을 분할하는 방법이고, 바이너리 트리 분할은, 수평선 또는 수직선을 이용하여 코딩 블록을 분할하는 방법으로 요약될 수 있다. 쿼드 트리 분할 및 바이너리 트리 분할되는 코딩 블록의 파티션 형태는 도 4 내지 도 8에 도시된 예에 한정되지 않으며, 도시된 것 이외의 확장된 파티션 형태가 사용될 수 있다. 즉, 코딩 블록은 도 4 내지 도 8에 도시된 것과 다른 형태로 재귀적으로 분할될 수 있다.

[0190] 도 11은 본 발명이 적용되는 다른 실시예로서, 비대칭형 쿼드 트리 분할이 허용되는 파티션 형태를 나타낸 도면이다.

[0191] 현재 블록이 쿼드 트리 분할되는 경우, 수평선 또는 수직선 중 적어도 하나는 코딩 블록을 비대칭 형태로 분할할 수도 있다. 여기서, 비대칭은, 수평선에 의해 분할된 블록들의 높이가 동일하지 않은 경우 또는 수직선에 의해 분할된 블록들의 너비가 동일하지 않은 경우 등을 의미할 수 있다. 일 예로, 수평선은 코딩 블록을 비대칭 형태로 분할함에 반해, 수직선은 코딩 블록을 대칭 형태로 분할할 수도 있고, 수평선은 코딩 블록을 대칭 형태로 분할함에 반해, 수직선은 코딩 블록을 비대칭 형태로 분할할 수도 있다. 또는, 수평선 및 수직선 모두 코딩 블록을 비대칭 형태로 분할할 수도 있다.

[0192] 도 11 (a)는 코딩 블록의 대칭형 쿼드 트리 분할 형태를 나타내고, (b)~(k)는 코딩 블록의 비대칭형 쿼드 트리

분할 형태를 나타낸 도면이다. 도 11 (a)는 수평선 및 수직선이 모두 대칭형 분할에 이용된 예를 나타낸 것이다. 도 11 (b) 및 (c)는 수평선은 대칭형 분할에 이용된 반면, 수직선은 비대칭형 분할에 이용된 예를 나타낸 것이다. 도 11 (d) 및 (e)는 수직선은 대칭형 분할에 이용된 반면, 수평선은 비대칭형 분할에 이용된 예를 나타낸 것이다.

[0193] 코딩 블록의 분할 형태를 특정하기 위해, 코딩 블록의 분할 형태와 관련된 정보를 부호화할 수 있다. 여기서, 상기 정보는, 코딩 블록의 분할 형태가 대칭형인지 또는 비대칭형인지를 나타내는 제1 지시자를 포함할 수 있다. 제1 지시자는 블록 단위로 부호화될 수도 있고, 수직선 또는 수평선 별로 부호화될 수 있다. 일 예로, 제1 지시자는 수직선이 대칭 분할에 이용되는지 여부를 나타내는 정보 및 수평선이 대칭 분할에 이용되는지 여부를 나타내는 정보를 포함할 수 있다.

[0194] 또는, 상기 제1 지시자는 수직선 또는 수평선 중 적어도 하나에 대해서만 부호화되고, 제1 지시자가 부호화되지 않는 다른 하나의 분할 형태는 제1 지시자에 의해 종속적으로 유도될 수도 있다. 예컨대, 제1 지시자가 부호화되지 않는 다른 하나의 분할 형태는 제1 지시자와 반대의 값을 가질 수 있다. 즉, 제1 지시자가 수직선이 비대칭분할에 이용됨을 나타내는 경우, 수평선은 제1 지시자와 반대인 대칭 분할에 이용되도록 설정될 수 있다.

[0195] 제1 지시자가 비대칭 분할임을 나타내는 경우, 수직선 또는 수평선에 대해 제2 지시자를 추가 부호화할 수도 있다. 여기서, 제2 지시자는, 비대칭 분할에 이용되는 수직선 또는 수평선의 위치 또는 수직선 또는 수평선에 의해 분할되는 블록 간의 비율 중 적어도 하나를 나타낼 수 있다.

[0196] 복수의 수직선 또는 복수의 수평선을 이용하여, 쿼드 트리 분할이 수행될 수도 있다. 일 예로, 하나 이상의 수직선 또는 하나 이상의 수평선 중 적어도 하나를 조합함으로써, 코딩 블록을 4개의 블록으로 분할하는 것도 가능하다.

[0197] 도 11 (f)~(k)는 복수의 수직선/수평선과 하나의 수평선/수직선을 조합함으로써, 코딩 블록을 비대칭적으로 분할하는 예를 나타낸 도면이다.

[0198] 도 11 (f)~(k)를 참조하면, 쿼드트리 분할은, 두개의 수직선 또는 두개의 수평선에 의해 코딩 블록을 세개의 블록으로 분할하고, 분할된 3개의 블록 중 어느 하나를 2개의 블록으로 분할함으로써 수행될 수 있다. 이때, 도 11 (f)~(k)에 도시된 예에서와 같이, 두개의 수직선 또는 두개의 수평선에 의해 분할된 블록 중 가운데에 위치한 블록이 하나의 수평선 또는 수직선에 의해 분할될 수 있다. 도시된 예에 그치지 않고, 코딩 블록의 일측 경계에 위치한 블록이 하나의 수평선 또는 수직선에 의해 분할될 수도 있다. 또는, 3개의 파티션 중 분할되는 파티션을 특정하기 위한 정보(예컨대, 파티션 인덱스)가 비트스트림을 통해 시그날링될 수도 있다.

[0199] 수평선 또는 수직선 중 적어도 하나는 코딩 블록을 비대칭 형태로 분할하는데 이용되고, 다른 하나는 코딩 블록을 대칭 형태로 분할하는데 이용될 수 있다. 일 예로, 복수의 수직선 또는 수평선이 코딩 블록을 대칭 형태로 분할하는데 이용되거나, 하나의 수평선 또는 수직선이 코딩 블록을 대칭 형태로 분할하는데 이용될 수 있다. 또는, 수평선 또는 수직선 모두 코딩 블록을 대칭 형태로 분할하는데 이용되거나, 비대칭 형태로 분할하는데 이용될 수도 있다.

[0200] 예를 들어, 도 11(f)는, 2개 수직선에 의해 비대칭 형태로 분할된 가운데 코딩 블록을 수평선에 의해 2개의 대칭형 코딩 블록으로 분할한 파티션 형태를 도시한 것이다. 또한, 도 11(g)는 2개 수평선에 의해 비대칭 형태로 분할된 가운데 코딩 블록을 수직선에 의해 2개의 대칭형 코딩 블록으로 분할한 파티션 형태를 도시한 것이다.

[0201] 반면, 도 11(h) 및 (i)는, 2개 수직선에 의해 비대칭 형태로 분할된 가운데 코딩 블록을 수평선에 의해 다시 2개의 비대칭형 코딩 블록으로 분할한 파티션 형태를 도시한 것이다. 또한, 도 11(j) 및 (k)는, 2개 수평선에 의해 비대칭 형태로 분할된 가운데 코딩 블록을 수직선에 의해 다시 2개의 비대칭형 코딩 블록으로 분할한 파티션 형태를 도시한 것이다.

[0202] 복수의 수직선/수평선과 하나의 수평선/수직선을 조합하는 경우, 코딩 블록은 적어도 2개의 서로 다른 크기로 구성된 4개의 파티션(즉, 4개의 코딩 블록)으로 분할된다. 이처럼 코딩 블록을 적어도 2개의 서로 다른 크기로 구성된 4개의 파티션으로 분할하는 것을 3종 비대칭 쿼드 트리 파티셔닝(Triple Type Asymmetric Quad-treeCU partitioning)이라 호칭할 수 있다.

[0203] 3종 비대칭 쿼드 트리 파티셔닝에 관한 정보는 전술한 제1 지시자 또는 제2 지시자 중 적어도 하나를 기초로 부호화될 수 있다. 일 예로, 제1 지시자는 코딩 블록의 분할 형태가 대칭형인지 또는 비대칭형인지를 나타낼 수 있다. 제1 지시자는 블록 단위로 부호화될 수도 있고, 수직선 또는 수평선 별로 부호화될 수도 있다. 일 예로,

제1 지시자는 하나 이상의 수직선이 대칭 분할에 이용되는지 여부를 나타내는 정보 및 하나 이상의 수평선이 대칭 분할에 이용되는지 여부를 나타내는 정보를 포함할 수 있다.

[0204] 또는, 상기 제1 지시자는 수직선 또는 수평선 중 적어도 하나에 대해서만 부호화되고, 제1 지시자가 부호화되지 않는 다른 하나의 분할 형태는 제1 지시자에 의해 종속적으로 유도될 수도 있다.

[0205] 제1 지시자가 비대칭 분할을 나타내는 경우, 수직선 또는 수평선에 대해 제2 지시자를 추가 부호화할 수도 있다. 여기서, 제2 지시자는, 비대칭 분할에 이용되는 수직선 또는 수평선의 위치 또는 수직선 또는 수평선에 의해 분할되는 블록 간의 비율 중 적어도 하나를 나타낼 수 있다.

[0206] 도 12는 본 발명이 적용되는 다른 실시예로서, 비대칭형 쿼드 트리 분할에 기반한 코딩 블록 분할 방법에 대한 흐름도이다.

[0207] 뎁스 k 코딩 블록을, 뎁스 k+1 코딩 블록으로 분할하는 것으로 가정한다. 우선, 뎁스 k 현재 블록에 쿼드 트리 분할이 적용되는 지를 판단한다(S1210). 상기 단계 S1210 판단결과, 쿼드 트리 분할이 적용되지 않았다면, 현재 블록은 분할 없이 뎁스 k+1 코딩 블록이 된다. 만약,단계 S1210 판단결과, 쿼드 트리 분할이 적용되었다면, 현재 블록에 비대칭 쿼드 트리 분할이 적용되는 지를 판단한다(S1220). 만약 비대칭형 쿼드 트리 분할이 적용되지 않고 대칭형 쿼드 트리 분할이 적용되었다면, 현재 블록을 4개의 정방형 블록으로 분할한다(S1230).

[0208] 반면, 만약 비대칭형 쿼드 트리 분할이 적용되었다면, 현재 블록에 3종 비대칭 쿼드 트리 분할이 적용되는 지를 판단한다(S1240). 만약 3종 비대칭 쿼드 트리 분할이 적용되지 않았다면, 현재 블록을 4개의 2종 비대칭 블록으로 분할한다(S1250). 이때 파티션 정보에 따라 도 11 (b)~(e) 중 어느 하나의 파티션 형태로 분할될 수 있다.

[0209] 반면, 만약 3종 비대칭 쿼드 트리 분할이 적용되었다면, 현재 블록을 4개의 3종 비대칭 블록으로 분할한다(S1260). 이때 파티션 정보에 따라 도 11 (f)~(k) 중 어느 하나의 파티션 형태로 분할될 수 있다.

[0210] 도 13은 본 발명이 적용되는 다른 실시예로서, 비대칭 쿼드 트리 분할이 적용되는 네트워크 추상화 계층 (NAL)에 포함되는 신택스 요소(syntax element)를 예를 들어 도시한 것이다. 본 발명이 적용되는 NAL 유닛은, 예를 들어, 비디오 파라미터 셋 (VPS), 시퀀스 파라미터 셋(SPS), 픽쳐 파라미터 셋(PPS) 및 적어도 하나 이상의 슬라이스 셋(Slice)을 포함할 수 있다.

[0211] 예를 들어, 도 13에서는 시퀀스 파라미터 셋(SPS)에 포함된 신택스 요소를 도시하였으나, 픽쳐 파라미터 셋(PPS) 또는 슬라이스 셋(Slice)에 신택스 요소를 포함하는 것도 가능하다. 또한, 신택스 요소별로 시퀀스 단위 또는 픽쳐 단위에 공통적으로 적용될 신택스 요소는 시퀀스 파라미터 셋(SPS) 또는 픽쳐 파라미터 셋(PPS)에 포함되도록 할 수 있다. 반면, 해당 슬라이스에만 적용되는 신택스 요소는 슬라이스 셋(Slice)에 포함되는 것이 바람직하다. 따라서, 이는 부호화 성능 및 효율을 고려하여 선택이 가능하다.

[0212] 신택스 요소 'Is_used_asymmertic_quad_tree_flag'는 쿼드 트리 분할이 비대칭으로 수행되는 지 여부를 나타낸다. 또한, 'Is_used_triple_asymmertic_quad_tree_flag'는 쿼드 트리 분할이 3종 비대칭으로 수행되는 지 여부를 나타낸다. 따라서, 만약 “Is_used_asymmertic_quad_tree_flag = 0” 이면 대칭 쿼드 트리 분할을 의미하므로, 'Is_used_triple_asymmertic_quad_tree_flag'는 시그날링 되지 않는다. 반면, “Is_used_asymmertic_quad_tree_flag = 1” 이고, “Is_used_triple_asymmertic_quad_tree_flag = 1” 이면 3종 비대칭 쿼드 트리 분할을 의미한다. 또한, “Is_used_asymmertic_quad_tree_flag = 1” 이고, “Is_used_triple_asymmertic_quad_tree_flag = 0” 이면 2종 비대칭 쿼드 트리 분할을 의미한다.

[0213] 신택스 요소 'hor_asymmetric_flag'는 비대칭 쿼드 트리 분할의 방향을 나타낸다. 즉, 상기 “Is_used_asymmertic_quad_tree_flag = 1” 인 경우, 수평 방향 또는 수직 방향으로의 비대칭 분할 여부를 나타낼 수 있다. 예를 들어, “hor_asymmetric_flag = 1” 이면 수평 방향으로 비대칭을 나타내고, “hor_asymmetric_flag = 0” 이면 수직 방향으로 비대칭을 나타낸다. 또한 다른 다른 대안으로, 'ver_asymmetric_flag'를 활용하는 것도 가능하다.

[0214] 신택스 요소 'width_left_asymmetric_flag'는 비대칭 쿼드 트리 분할의 또 다른 방향을 나타낸다. 즉, 상기 “Is_used_asymmertic_quad_tree_flag = 1” 인 경우, 너비 방향 좌측 또는 우측 방향으로의 비대칭 분할 여부를 나타낼 수 있다. 예를 들어, “width_left_asymmetric_flag' = 1” 이면 너비 좌측 방향으로 비대칭을 나타내고, “width_left_asymmetric_flag = 0” 이면 너비 우측방향으로 비대칭을 나타낸다.

[0215] 또한 신택스 요소 'height_top_asymmetric_flag'는 비대칭 쿼드 트리 분할의 또 다른 방향을 나타낸다. 즉, 상기 “Is_used_asymmertic_quad_tree_flag = 1” 인 경우, 높이 방향 상측 또는 하측 방향으로의 비대칭 분할

여부를 나타낼 수 있다. 예를 들어, “height_top_asymmetric_flag' = 1” 이면 높이 상측 방향으로 비대칭을 나타내고, “height_top_asymmetric_flag = 0” 이면 높이 하측 방향으로 비대칭을 나타낸다.

[0216] 또한, 신택스 요소 'is_used_symmetric_line_flag'는 3종 비대칭 쿼드 트리 분할의 경우, 가운데 블록에 대한 대칭 블록 여부를 나타낸다. 즉, 상기 “Is_used_asymmertic_quad_tree_flag = 1” 및 “Is_used_triple_asymmertic_quad_tree_flag = 1” 인 경우, 가운데 블록의 대칭 분할 여부를 나타낸다.

[0217] 따라서, 상기 신택스 요소들의 조합을 통해, 도 11 (a)~(k) 에 도시된 파티션 형태를 표현하는 것이 가능하다. 예를 들어, “Is_used_asymmertic_quad_tree_flag = 0” 이면 도 11(a) 파티션 형태와 같이 4개 대칭 블록으로 분할됨을 의미한다.

[0218] 또한, “Is_used_asymmertic_quad_tree_flag = 1” 이고 “Is_used_triple_asymmertic_quad_tree_flag = 0” 이면, 도 11 (b)~(e) 파티션 형태중 어느 하나에 해당된다. 이 경우, “hor_asymmetric_flag = 1” 이고 “width_left_asymmetric_flag' = 1” 이면 도 11 (b) 파티션 형태를 의미한다. 또한, “hor_asymmetric_flag = 1” 이고 “width_left_asymmetric_flag' = 0” 이면 도 11 (c) 파티션 형태를 의미한다. 또한, “hor_asymmetric_flag = 0” 이고 “height_top_asymmetric_flag' = 1” 이면 도 11 (d) 파티션 형태를 의미한다. 또한, “hor_asymmetric_flag = 0” 이고 “height_top_asymmetric_flag' = 0” 이면 도 11 (e) 파티션 형태를 의미한다.

[0219] 또한, “Is_used_asymmertic_quad_tree_flag = 1” 이고 “Is_used_triple_asymmertic_quad_tree_flag = 1” 이면, 도 11 (f)~(k) 파티션 형태중 어느 하나에 에 해당된다. 이 경우, “is_used_symmetric_line_flag = 1” 이면, 도 11 (f),(g) 파티션 형태중 어느 하나에 해당되고, “is_used_symmetric_line_flag = 0” 이면, 도 11 (h)~(k) 파티션 형태중 어느 하나에 해당된다. 또한, 상기 “is_used_symmetric_line_flag = 1” 이고, “hor_asymmetric_flag = 1” 이면 도 11 (f) 파티션 형태로 정의하고, “hor_asymmetric_flag 0” 이면 도 11 (g) 파티션 형태로 정의할 수 있다.

[0220] 또한, “Is_used_asymmertic_quad_tree_flag = 1”, “Is_used_triple_asymmertic_quad_tree_flag = 1” 및 “is_used_symmetric_line_flag = 0” 인 경우에는, “hor_asymmetric_flag”, “width_left_asymmetric_flag” 및 “height_top_asymmetric_flag” 에 의해 파티션 형태를 정의할 수 있다. 예를 들어, “hor_asymmetric_flag = 1” 이고 “height_top_asymmetric_flag= 0” 이면, 도 11 (h) 파티션 형태를 의미한다. 또한, “hor_asymmetric_flag = 1” 이고 “height_top_asymmetric_flag= 1” 이면, 도 11 (i) 파티션 형태를 의미한다. 또한, “hor_asymmetric_flag = 0” 이고 “width_left_asymmetric_flag' = 0” 이면 도 11 (j) 파티션 형태를 의미한다. 또한, “hor_asymmetric_flag = 0” 이고 “width_left_asymmetric_flag' = 1” 이면 도 11 (k) 파티션 형태를 의미한다.

[0221] 또한, 다른 대안으로, 'asymmetric_quadtree_partition_index'에 의해 상기 도 11(a)~(k) 파티션 형태를 각각 인덱스로 표시하는 것도 가능하다.

[0222] 도 14는 본 발명이 적용되는 또 다른 실시예로서, 쿼드 트리 및 트리플 트리 분할이 허용되는 파티션 형태를 나타낸 도면이다.

[0223] 코딩 블록은 쿼드 트리(quad tree)와 트리플 트리(triple tree) 중 적어도 하나에 기초하여 계층적으로 분할될 수 있다. 여기서, 쿼드 트리 기반의 분할은 2Nx2N 코딩 블록이 4개의 NxN 코딩 블록으로 분할되는 방식(도 14(a))을, 트리플 트리 기반의 분할은 하나의 코딩 블록이 3개의 코딩 블록으로 분할되는 방식을 각각 의미할 수 있다. 트리플 트리 기반의 분할이 수행되었다 하더라도, 하위 뎁스에서는 정방형인 코딩 블록이 존재할 수 있다.

[0224] 트리플 트리 기반의 분할은 대칭적으로 수행될 수도 있고 (도 14(b)), 비대칭적으로 수행될 수도 있다 (도 14(c)). 또한, 트리플 트리 기반으로 분할된 코딩 블록은 정방형 블록일 수도 있고, 직사각형과 같은 비정방형 블록일 수도 있다. 일 예로, 트리플 트리 기반의 분할이 허용되는 파티션 형태는 도 14 (b)에 도시된 예에서와 같이, 너비 또는 높이가 동일한 대칭형(symmetric)인 2Nx(2N/3) (수평 방향 비 정방 코딩 유닛) 또는 (2N/3)x2N (수직 방향 비정방 코딩 유닛)이 될 수 있다. 또한, 일 예로, 트리플 트리 기반의 분할이 허용되는 파티션 형태는 도 14 (c)에 도시된 예에서와 같이, 적어도 너비 또는 높이가 상이한 코딩 블록을 포함하는 비대칭형(asymmetric) 파티션 형태가 될 수 있다. 예를 들어, 도 14 (c)에 의한 비대칭형 트리플 트리 파티션 형태는, 적어도 2개의 코딩 블록(1401, 1403)은 동일한 너비 (또는 높이) 크기로 k값을 가지고 양측에 위치하도록 정의하고, 나머지 하나의 블록(1402)은 너비 (또는 높이) 크기로 2k 값을 가지며 상기 동일 크기 블록들 (1401,

1403) 사이에 위치하도록 정의할 수 있다.

[0225] 관련하여, CTU 또는 CU를 도 14에 도시한 바와 같이 비 정방 형태인 3개의 서브 파티션로 나누는 방식을, 트리플 트리 파티셔닝 방법(triple tree CU partitioning)이라고 부른다. 트리플 트리 파티셔닝으로 나뉘어진 CU는 추가적으로 파티셔닝을 수행하지 않도록 제한할 수도 있다.

[0226] 도 15는 본 발명이 적용되는 또 다른 실시예로서, 쿼드 트리 및 트리플 트리 분할 에 기반한 코딩 블록 분할 방법에 대한 흐름도이다.

[0227] 뎁스 k 코딩 블록을, 뎁스 k+1 코딩 블록으로 분할하는 것으로 가정한다. 우선, 뎁스 k 현재 블록에 쿼드 트리 분할이 적용되는 지를 판단한다(S1510). 만약 쿼드 트리 분할이 적용되었다면, 현재 블록을 4개의 정방형 블록으로 분할한다(S1520). 반면, 만약 쿼드 트리 분할이 적용되지 않았다면, 현재 블록에 트리플 트리 분할이 적용되는 지를 판단한다(S1530). 만약 트리플 트리 분할도 적용되지 않았다면, 현재 블록은 분할 없이 뎁스 k+1 코딩 블록이 된다.

[0228] 상기 S1530 판단 결과, 현재 블록에 트리플 트리 분할이 적용되었다면, 대칭형 트리플 분할 또는 비대칭형 트리플 분할 중 어느 방식이 적용되는 지를 확인한다(S1540). 상기 S1540 판단 결과에 따라, 현재 블록에 적용되는 파티션 형태를 결정한다(S1550). 예를 들어, 상기 S1550 단계에 적용되는 파티션 형태는, 대칭형인 경우 도 14(b) 형태중 어느 하나가 적용되고, 비대칭형인 경우 도 14(c) 형태 중 어느 하나가 될 수 있다. 상기 단계 S1550을 통해, 결정된 파티션 형태에 따라, 현재 블록을 3개의 뎁스 k+1 코딩 블록으로 분할하게 된다(S1560).

[0229] 도 16은 본 발명이 적용되는 또 다른 실시예로서, 쿼드 트리 및 트리플 트리 분할이 적용되는 네트워크 추상화 계층 (NAL)에 포함되는 신택스 요소(syntax element)를 예를 들어 도시한 것이다. 본 발명이 적용되는 NAL 유닛은, 예를 들어, 비디오 파라미터 셋 (VPS), 시퀀스 파라미터 셋(SPS), 픽처 파라미터 셋(PPS) 및 적어도 하나 이상의 슬라이스 셋(Slice)을 포함할 수 있다.

[0230] 예를 들어, 도 16에서는 시퀀스 파라미터 셋(SPS)에 포함된 신택스 요소를 도시하였으나, 픽처 파라미터 셋(PPS) 또는 슬라이스 셋(Slice)에 신택스 요소를 포함하는 것도 가능하다. 또한, 신택스 요소별로 시퀀스 단위 또는 픽처 단위에 공통적으로 적용될 신택스 요소는 시퀀스 파라미터 셋(SPS) 또는 픽처 파라미터 셋(PPS)에 포함되도록 할 수 있다. 반면, 해당 슬라이스에만 적용되는 신택스 요소는 슬라이스 셋(Slice)에 포함되는 것이 바람직하다. 따라서, 이는 부호화 성능 및 효율을 고려하여 선택이 가능하다.

[0231] 신택스 요소 'quad_split_flag'는 코딩 블록이 4개의 코딩 블록으로 분할되는지 여부를 나타낸다. 'triple_split_flag'는 코딩 블록이 3개의 코딩 블록으로 분할되는지 여부를 나타낼 수 있다. 코딩 블록이 3개의 코딩 블록으로 분할되는 경우, 코딩 블록의 분할 방향이 수직 방향인지 또는 수평 방향인지 여부를 나타내는 'is_hor_split_flag'가 시그널링될 수 있다. “is_hor_split_flag = 1” 이면 수평 방향을 “is_hor_split_flag = 0” 이면 수직방향을 나타내는 것으로 정의할 수 있다.

[0232] 또한, 다른 대안으로, 'isUseTripleTreeFlag'를 통해 현재 블록에 트리플 트리 파티셔닝이 적용 여부를 나타내고, 또한, 코딩 블록의 분할 방향을 나타내는 신택스 요소로서, 'hor_triple_flag' 는 코딩 블록이 수평 방향으로 분할되었는지 여부를 나타낼 수 있다. 예를 들어, “hor_triple_flag = 1”인 경우, 코딩 블록이 수평 방향으로 분할됨을 나타내고, “hor_triple_flag = 0”인 경우, 코딩 블록이 수직 방향으로 분할됨을 나타낼 수 있다. 또는 'hor_triple_flag' 대신 코딩 블록이 수직 방향으로 분할되었는지 여부를 나타내는 ver_triple_flag 가 이용하여 동일한 방식으로 설정할 수 있다.

[0233] 또한, 비대칭 트리플 트리 파티셔닝이 허용되는지 여부를 나타내는 신택스 요소로서, 'asymmetric_triple_tree_flag'을 정의할 수 있다. 예를 들어, “asymmetric_triple_tree_flag = 1” 이면 비대칭 트리플 트리 파티셔닝이 사용되었음을 나타내고, “asymmetric_triple_tree_flag = 0” 이면, 비대칭 트리플 트리 파티셔닝이 사용되지 않았음을 나타낼 수 있다. 반면, 픽처 단위 또는 슬라이스 단위로 트리플 트리 파티셔닝이 사용되지 않는 경우, 'asymmetric_triple_tree_flag'를 시그널링하지 않고, 그 값을 0으로 설정할 수도 있다.

[0234] 따라서, 상기 신택스 요소들의 조합을 통해, 도 14 (a)~(c) 에 도시된 파티션 형태를 표현하는 것이 가능하다. 예를 들어, “isUseTripleTreeFlag = 0” 이면 도 14(a) 파티션 형태와 같이 4개 대칭 블록으로 분할됨을 의미한다.

[0235] 또한, “isUseTripleTreeFlag = 1” 이고 “asymmetric_triple_tree_flag = 0” 이면, 도 14 (b) 파티션 형태

중 어느 하나에 해당된다. 이때, “hor_triple_flag = 1”이면, 도 14 (b) (2N/3)x2N 파티션 형태를 의미하는 것으로 정의하고. “hor_triple_flag = 0”이면, 도 14 (b) 2Nx(2N/3) 파티션 형태를 의미하는 것으로 정의할 수 있다.

[0236] 또한, “isUseTripleTreeFlag = 1” 이고 “asymmetric_triple_tree_flag = 1” 이면, 도 14 (c) 파티션 형태 중 어느 하나에 해당된다. 이때, “hor_triple_flag = 1”이면, 도 14 (c) 왼쪽 파티션 형태를 의미하는 것으로 정의하고. “hor_triple_flag = 0”이면, 도 14 (c) 오른쪽 파티션 형태를 의미하는 것으로 정의할 수 있다.

[0237] 또한, 다른 대안으로, 'asymmetric_tripletree_partition_index'에 의해 상기 도 14(a)~(c) 파티션 형태를 각각 인덱스로 표시하는 것도 가능하다.

[0238] 도 17은 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할이 허용되는 파티션 형태를 나타낸 도면이다.

[0239] 전술한 쿼드 트리 파티셔닝, 바이너리 파티셔닝, 또는 트리플 트리 파티셔닝중 적어도 어느 하나를 이용하여 CTU 또는 CU를 파티셔닝하는 방법을 멀티 트리 파티셔닝(multi tree CU partitioning)이라고 부른다. 전술한 예시 중 어느 N개의 파티션을 사용하여 CTU 또는 CU를 파티셔닝할 수 있다. 구체적으로 예를 들어, 도 17과 같이 9개의 파티셔닝을 이용하여 CTU 또는 CU를 파티셔닝할 수 있다.

[0240] 시퀀스 단위 또는 픽쳐 단위로 쿼드 트리 파티셔닝, 바이너리 트리 파티셔닝, 또는 트리플 트리 파티셔닝 모두를 사용하여 파티셔닝을 하거나, 그 중 어느 하나 또는 어느 두개의 파티셔닝을 사용하여 CTU 또는 CU를 파티셔닝 할 수도 있다.

[0241] 쿼드 트리 파티셔닝은 기본으로 사용하고, 바이너리 트리 파티셔닝과 트리플 트리 파티셔닝은 선택적으로 사용할 수도 있다. 이 때, 시퀀스 헤더(sequence parameter set) 또는 픽쳐 헤더(picture parameter set)에서 바이너리 트리 파티셔닝을 사용하는지 및/또는 트리플 트리 파티셔닝을 사용하는지를 시그날링할 수 있다.

[0242] 또는 쿼드 트리 파티셔닝과 트리플 트리 파티셔닝은 기본으로 사용하고, 바이너리 트리 파티셔닝은 선택적으로 사용할 수도 있다. 예를 들어, 시퀀스 헤더에서 바이너리 트리 파티셔닝을 사용하는지 여부를 나타내는 신택스 isUseBinaryTreeFlag을 시그날링할 수 있다. isUseBinaryTreeFlag값이 1이면 현재 시퀀스에서 바이너리 트리 파티셔닝을 사용하여 CTU 또는 CU를 파티셔닝할 수 있다. 시퀀스 헤더에서 트리플 트리 파티셔닝을 사용하는지 여부를 나타내는 신택스 isUseTripleTreeFlag을 시그날링할 수도 있다. isUseTripleTreeFlag값이 1이면 현재 시퀀스 헤더에서 트리플 트리 파티셔닝을 사용하여 CTU 또는 CU를 파티셔닝할 수 있다.

[0243] 멀티 트리 파티셔닝에 의해 분할된 파티션 형태는, 예를 들어, 도 17 (a)~(i)에 도시된 9개 기본 파티션으로 한정할 수 있다. 도 17 (a)는 쿼드 트리 파티션 형태를 나타내고, (b)~(c)는 대칭형 바이너리 트리 파티션 형태를 나타내고, (d)~(e)는 비대칭형 트리플 트리 파티션 형태를 나타내고, (f)~(i)는 비대칭형 바이너리 트리 파티션 형태를 나타낸다. 관련하여 도 17에 도시된 각 파티션 형태에 대해서는 전술한 바와 동일하여 이하 상세한 설명은 생략한다.

[0244] 또한, 다른 대안으로, 멀티 트리 파티셔닝에 의해 분할된 파티션 형태로서, 예를 들어, 도 18 (j)~(u)에 도시된 12개의 파티션을 더 포함하는 것으로 확장할 수 있다. 도 18 (j)~(m)은 비대칭 쿼드 트리 파티션 형태를 나타내고, (n)~(s)는 3종 비대칭 쿼드 트리 파티션 형태를 나타내고, (t)~(u)는 대칭형 트리플 트리 파티션 형태를 나타내다. 관련하여 도 18에 도시된 각 파티션 형태에 대해서는 전술한 바와 동일하여 이하 상세한 설명은 생략한다.

[0245] 도 19는 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할에 기반한 코딩 블록 분할 방법에 대한 흐름도이다.

[0246] 뎁스 k 코딩 블록을, 뎁스 k+1 코딩 블록으로 분할하는 것으로 가정한다. 우선, 뎁스 k 현재 블록에 쿼드 트리 분할이 적용되는 지를 판단한다(S1910). 만약 쿼드 트리 분할이 적용되지 않았다면, 현재 블록에 바이너리 트리 분할이 적용되는 지를 판단한다(S1950). 또한, 만약 바이너리 트리 분할이 적용되지 않았다면, 현재 블록에 트리플 트리 분할이 적용되는 지를 판단한다(S1990). 만약 상기 단계 S1950 판단결과, 트리플 트리 분할도 적용되지 않았다면, 현재 블록은 분할 없이 뎁스 k+1 코딩 블록이 된다.

[0247] 여기서, 상기 단계 S1910 판단 결과, 만약 쿼드 트리 분할이 적용되었다면, 대칭 또는 비대칭 쿼드 트리 분할 여부를 확인한다(S1920). 이후, 파티션 정보를 확인하여 현재 블록의 블록 파티션 형태를 결정하고(S1930), 결정된 파티션 형태에 따라 현재 블록을 4개의 블록으로 분할한다(S1940). 예를 들어, 대칭형 쿼드 트리가 적용된 경우에는, 도 17 (a) 파티션 형태로 분할한다. 또한, 비대칭형 쿼드 트리가 적용된 경우에는, 도 18 (j)~(m) 중

어느 하나의 파티션 형태로 분할한다. 또는, 3종 비대칭형 쿼드 트리가 적용된 경우에는, 도 18 (n)~(s) 중 어느 하나의 파티션 형태로 분할한다. 단, 전술한 바와 같이, 만약 멀티 트리 파티션 형태를 도 17의 기본 파티션 형태만 적용하는 경우에는, 쿼드 트리의 비대칭 여부를 판단하지 않고, 도 17 (a)의 대칭 정방형 블록만 적용할 수 있다.

[0248] 또한, 상기 단계 S1950 판단 결과, 만약 바이너리 트리 분할이 적용되었다면, 대칭 또는 비대칭 바이너리 트리 분할 여부를 확인한다(S1960). 이후, 파티션 정보를 확인하여 현재 블록의 블록 파티션 형태를 결정하고(S1970), 결정된 파티션 형태에 따라 현재 블록을 2개의 블록으로 분할한다(S1980). 예를 들어, 대칭형 바이너리 트리가 적용된 경우에는, 도 17 (b) 및 (c) 중 어느 하나의 파티션 형태로 분할한다. 또한, 비대칭형 바이너리 트리가 적용된 경우에는, 도 17 (f)~(i) 중 어느 하나의 파티션 형태로 분할한다.

[0249] 또한, 상기 단계 S1990 판단 결과, 만약 트리플 트리 분할이 적용되었다면, 대칭 또는 비대칭 트리플 트리 분할 여부를 확인한다(S1960). 이후, 파티션 정보를 확인하여 현재 블록의 블록 파티션 형태를 결정하고(S1970), 결정된 파티션 형태에 따라 현재 블록을 3개의 블록으로 분할한다(S1980). 예를 들어, 비대칭형 트리플 트리가 적용된 경우에는, 도 17 (d) 및 (e) 중 어느 하나의 파티션 형태로 분할한다. 또한, 대칭형 바이너리 트리가 적용된 경우에는, 도 18 (t)~(u) 중 어느 하나의 파티션 형태로 분할한다. 단, 전술한 바와 같이, 만약 멀티 트리 파티션 형태를 도 17의 기본 파티션 형태만 적용하는 경우에는, 트리플 트리의 비대칭 여부를 판단하지 않고, 17 (d) 및 (e)의 기 정의된 비대칭 트리플 블록만 적용할 수 있다.

[0250] . 멀티 트리 파티셔닝을 표현하는 신택스 요소로서, 멀티 트리 분할 여부를 나타내는 'is_used_Multitree_flag'를 정의할 수 있다. 또한, 전술한 도 10, 13 및 16에서 도시되고 설명된 신택스 요소들을 멀티 트리 파티셔 형태를 결정하는 정보로 활용하는 것이 가능하다.

[0252] 도 20 및 도 21은 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할에 의해 분할된 코딩 블록의 병렬 처리 방법을 설명하기 위해 도시한 것이다. 예를 들어, 도 20은 멀티 트리 분할에 의해 분할된 제1타입 코딩 블록의 병렬 처리 방법을, 도 21은 멀티 트리 분할에 의해 분할된 제2타입 코딩 블록의 병렬 처리 방법을 설명하기 위해 도시한 것이다. 상기 제1 타입 코딩 블록은, 병렬 처리 블록의 너비(w) 크기가 높이(h) 크기 보다 작은 경우로서, 수직 방향으로 더 긴 길이를 가지는 비정방형 블록을 의미한다. 또한, 상기 제2 타입 코딩 블록은, 병렬 처리 블록의 높이(h) 크기가 너비(w) 크기 보다 작은 경우로서, 수평 방향으로 더 긴 길이를 가지는 비정방형 블록을 의미한다.

[0253] 도 20은 예를 들어, 코딩 블록(CU)이 하위 뎁스 코딩 블록 (CU0 ~ CU9)로 분할된 경우를 도시한 것이다. 예를 들어, 분할된 CU0 및 CU1의 크기는 2x4, 분할된 CU2, CU3 및 CU4의 크기는 4x4, 분할된 CU5 및 CU6의 크기는 2x8, 분할된 CU7의 크기는 4x8, 분할된 CU8 및 CU9의 크기는 8x8 로 분할된 경우를 도시한 것이다.

[0254] 또한, 도 21은 예를 들어, 코딩 블록(CU)이 하위 뎁스 코딩 블록 (CU10 ~ CU17)로 분할된 경우를 도시한 것이다. 예를 들어, 분할된 CU10 및 CU11의 크기는 8x2, 분할된 CU12, CU15 및 CU16의 크기는 8x4, 분할된 CU13 및 CU14의 크기는 4x8, 분할된 CU17의 크기는 8x8로 분할된 경우를 도시한 것이다.

[0255] 단, 상기 도 20 및 도 21의 분할은 전술한 멀티 트리 파티셔닝 방법 중 하나를 적용하여 분할한 것을 도시한 것일 뿐, 파티션 방식에 따라서 도 20 및 도 21과 상이한 다양한 파티션 형태가 가능하다.

[0256] 관련하여, 예를 들어, 도 20에 도시된 CU1 (2002)이 인트라 예측 모드로 부호화 되는 경우에는 종래 HEVC나 기존 레거시 코덱에서는 CU0 (2001)를 부호화 한 후에 CU1 (2002)을 부호화할 수 있다. 즉, 종래에는 CU0 (2001)에서 복원된 영상 또는 샘플들을 CU1 (2002)의 인트라 레퍼런스 샘플 (즉, 참조 샘플)로 사용하기 때문에 CU0 (2001)와 CU1 (2002)을 동시에 부호화 할 수 없게 된다. 또한, 마찬가지로, 종래에는 도20에 도시된 CU6 (2004)을 인트라 예측 모드로 부호화 하기 위해서는, CU5 (2003)를 부호화 한 후에 CU6 (2004)을 부호화 할 수 있었다. 즉, 종래에는 CU5 (2003)에서 복원된 영상 또는 샘플들을 CU6 (2004)의 인트라 레퍼런스 샘플 (즉, 참조 샘플)로 사용하기 때문에 CU5 (2003)와 CU6 (2004)을 동시에 부호화 할 수 없게 된다.

[0257] 전술한 본 발명의 멀티 트리 파티셔닝을 적용시, 2x16, 16x2 또는 2x8, 8x2 같이 너비 또는 높이가 작은 비 정방 형태 코딩 유닛을 사용할 수 있어서 작은 오브젝트나 세밀한 오브젝트를 효과적으로 압축할 수 있는 장점이 있다. 하지만, 멀티 트리 파티셔닝을 적용시 다수의 비정방 코딩 유닛의 생성으로 인한 데이터 디펜던시(data dependency) 문제로, 하드웨어 또는 소프트웨어 구현 시에 대기 시간 (latency)이 길어져 구현 복잡도 및 속도를 증가시키는 원인이 될 수 있다. 따라서, 본 발명은 전술한 멀티 트리 파티셔닝 방식을 더욱 효과적으로 지

원하기 위해, 다수의 독립 코딩 블록을 병렬 처리하는 영상 복호화 및 부호화 방법을 제안한다.

[0258] 구체적으로, 본 발명에 의한 병렬 처리 방법에 의해, 현재 CU는 이웃 CU와 독립적으로 인트라 또는 인터 예측을 수행할 수 있다. 현재 CU는 이웃 CU의 텍스쳐 데이터 (예를 들어, 복원 샘플, 잔차 샘플, 예측 샘플 등) 및/또는 부호화 파라미터를 공유하는 데이터디펜던시(dependency)없이 예측을 수행할 수 있다. 여기서, 독립적 예측이라 함은, 이웃 CU와의 데이터를 이용하지 않음을 의미할 수도 있고, 구현 방식에 따라서 현재 CU와 이웃 CU를 병렬적으로 예측을 수행함을 의미할 수도 있다. 또는, 해당 이웃 CU는 현재 CU의 부호화/복호화를 위한 비가용 블록으로 설정될 수 있다.

[0259] 또한, 멀티 트리 파티셔닝으로 코딩 유닛이 파티션된 경우에 코딩 유닛의 너비 또는 높이 중 적어도 하나에 기초하여 현재 CU와 이웃 CU를 독립적으로 부호화/복호화할 수도 있다. 이웃 CU의 개수는 하나 또는 그 이상일 수 있다. 이웃 CU의 위치는 현재 CU에 수평, 수직 또는 대각선 방향 중 적어도 하나의 방향으로 연속한 것일 수 있다. 또한, 현재 CU의 예측 모드 및/또는 인트라 예측 모드의 방향성, 상위 CU 내에서 현재 CU의 위치, 현재 CU의 모양, 크기 관련 정보, 또는 휘도/색차 성분 관련 정보 중 적어도 하나에 기반하여 이웃 CU가 특정될 수 있다.

[0260] 인접한 복수의 코딩 유닛이 독립적인 병렬 처리 가능한 CU인지 여부는 다양한 기준에 의해 설정할 수 있다. 예를 들어, 코딩 유닛의 너비(w) 또는 높이(h)가 특정 임계값보다 작은지 여부에 기초하여 복수의 코딩 유닛을 독립적으로 부호화/복호화할 수 있다. 또는 코딩 유닛의 너비(w)와 높이(h) 비 (즉, w/h 또는 h/w)값 특정 임계값보다 작은지 여부에 기초하여 복수의 코딩 유닛을 독립적으로 부호화/복호화할 수도 있다. 또는, 코딩 유닛의 너비와 높이의 합 또는 코딩 유닛의 샘플 개수가 특정 임계값보다 작은지 여부에 기초하여 복수의 코딩 유닛을 독립적으로 부호화/복호화할 수 있다. 상기 특정 임계값은 부호화기/복호화기에 기 설정된 고정된 값일 수도 있고, 부호화기에서 최적의 임계값을 결정하고, 이를 부호화하여 복호화기로 시그날링할 수도 있다. 이때 부호화된 임계값 관련 정보는 시퀀스, 픽쳐, 슬라이스, 블록 중 적어도 하나의 단위에서 시그날링될 수 있다.

[0261] 구체적으로 예를 들어, 도 20의 {CU0, CU1}, {CU5, CU6} 및 도 21의 {CU10, CU11}와 같이 코딩 유닛의 너비 또는 높이가 4 보다 작은 경우에 두 개의 연속된 코딩 유닛을 독립적으로 부호화/복호화할 수 있다.

[0262] 여기서, 코딩 스캔 순서상 앞에 있는 CU를 대응 CU (corresponding CU)라하고, 코딩 스캔 순서상 뒤에 있는 CU를 대상 CU (target CU)라 정의한다. 또는, 함께 병렬 처리되는 복수의 블록들 중, 현재 코딩 블록을 현재 병렬 처리 블록으로 명하고, 현재 병렬 처리 블록에 인접한 코딩 블록을 이웃 병렬 처리 블록으로 명명할 수 있다. 즉, 도 20을 참조하면, CU0 (2001) 및 CU5 (2003)은 대응 CU에 해당되고, CU1 (2002) 및 CU6 (2004)은 대상 CU에 해당된다. 또한, 도 21을 참조하면, CU10 (2101)은 대응 CU에 해당되고, CU11 (2102)은 대상 CU에 해당된다.

[0263] 또한, 구체적으로, 현재 병렬 처리 블록인 대상 CU (예를 들어, CU1, CU6, CU11)를 병렬 처리를 하기 위해서는, 대상 CU 복호화/부호화를 위한 참조 샘플을 획득하는 것이 필요하다. 본 발명에 의하면, 대상 CU 참조 샘플은 아직 복원되지 않은 이웃 병렬 처리 블록인 대응 CU (예를 들어, CU0, CU5, CU10) 영역에서 유도하지 않고, 대응 CU에서 사용한 참조 샘플을 활용하는 것을 특징으로 한다. 또는 대상 CU 참조 샘플은, 대상 CU 기준점 샘플 값과 대응 CU 기준점 샘플 값 간의 차이를 활용하여 유도 하는 것을 특징으로 한다.

[0264] 여기서, 상기 기준점 샘플은 대상 CU 또는 대응 CU에 인접한 하나 또는 그 이상의 샘플일 수 있다. 또한, 상기 기준점 샘플은 대상 CU 또는 대응 CU의 상단, 좌측 또는 좌-상단 코너에 인접한 샘플일 수 있다. 상기 기준점 샘플 전부 또는 일부의 위치는 대상 CU 또는 대응 CU의 크기, 형태, 가로 세로비, 부호화/복호화 대상 샘플의 위치 등에 따라 가변적으로 특정될 수도 있다. 또는 대상 CU 또는 대응 CU와 관계없이 기 고정된 위치의 샘플이 이용될 수도 있다.

[0265] 또한, 상기 차이는 대상 CU의 기준점 샘플 값과 대응 CU의 기준점 샘플 값 간의 차분을 통해 유도될 수도 있고, 각 기준점 샘플 값에 소정의 가중치를 적용한 후 차분을 통해 유도될 수도 있다. 또는, 상기 차이는 차분을 통해 생성된 값을 소정의 스케일링 팩터로 스케일링하여 유도될 수도 있다. 또는 상기 차이는 차분을 통해 생성된 값과 매핑된 오프셋으로 유도될 수도 있다. 관련하여, 대상 CU에 대한 병렬 처리 부호화/복호화를 수행하기 위한 참조 샘플을 획득하는 구체적인 방법에 대해서는, 도26~도29에서 상세히 후술하고자 한다.

[0266] 도 22는 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할에 의해 분할된 병렬 처리 가능한 코딩 블록을 설명하기 위해 도시한 것이다. 멀티 트리 파티셔닝으로 코딩 유닛이 파티션된 경우에 코딩 유닛의 너비 또는 높이 중 적어도 하나에 기초하여 현재 CU와 이웃 CU를 독립적으로 병렬 처리하는 것이 가능하다. 또한. 이웃 병렬

처리 CU의 위치는 현재 병렬 처리 CU에 수평, 수직 또는 대각선 방향 중 적어도 하나의 방향으로 연속한 것일 수 있다. 예를 들어, 도 22 (a)는 수평 방향 병렬 처리 블록을, 도 22 (b)는 수직 방향 병렬 처리 블록을, 도 22 (c)는 대각선 방향 병렬 처리 블록을, 도 22 (d)는 수평, 수직, 대각선 방향 병렬 처리 블록이 모두 포함된 경우를, 각각 도시한 것이다.

[0267] 여기서, 상기 병렬 처리 블록의 결정은, 해당 코딩 유닛의 부호화/복호화 성능 및 효율에 따라 부호화/복호화시 각각 선택하는 것이 가능하다. 또는, 상기 병렬 처리 블록의 결정은, 해당 코딩 유닛의 부호화 성능 및 효율에 따라 부호화시 선택하고 선택된 병렬 처리 정보를 복호화기로 시그날링하여 전송함에 의해, 복호화시 해당 결정된 블록을 병렬 처리하는 것이 가능하다.

[0268] 도 22 (a)는, 예를 들어, 코딩 유닛이 하위 뎁스 코딩 유닛 (CU18~CU29)로 분할된 경우를 도시한 것이다. 여기서, 전술한 다양한 병렬 처리 블록의 결정 방식중 인접한 수평방향 코딩 유닛의 너비 또는 높이 크기에 근거하여, 세번째 뎁스에 의해 분할된 {CU27, CU28}를 수평방향 인접한 병렬 처리 블록으로 결정할 수 있다. 또한, 동일한 기준에 의해, 다른 위치의 세번째 뎁스에 의해 분할된 {CU18, CU19} 및 {CU21, CU22}를 각각 수평방향 인접한 병렬 처리 블록으로 결정할 수 있다.

[0269] 도 22 (b)는, 예를 들어, 코딩 유닛이 하위 뎁스 코딩 유닛 (CU30~CU40)로 분할된 경우를 도시한 것이다. 여기서, 전술한 다양한 병렬 처리 블록의 결정 방식중 인접한 수직방향 코딩 유닛의 너비 또는 높이 크기에 근거하여, 세번째 뎁스에 의해 분할된 {CU36, CU37}를 수직방향 인접한 병렬 처리 블록으로 결정할 수 있다. 또한, 동일한 기준에 의해, 다른 위치의 세번째 뎁스에 의해 분할된 {CU30, CU31}을 각각 수평방향 인접한 병렬 처리 블록으로 결정할 수 있다.

[0270] 도 22 (c)는, 예를 들어, 코딩 유닛이 하위 뎁스 코딩 유닛 (CU41~CU54)로 분할된 경우를 도시한 것이다. 여기서, 전술한 다양한 병렬 처리 블록의 결정 방식중 인접한 대각선 방향 코딩 유닛의 너비 또는 높이 크기에 근거하여, 세번째 뎁스에 의해 분할된 {CU41, CU44}을 대각선 방향 인접한 병렬 처리 블록으로 결정할 수 있다. 또한, 동일한 기준에 의해, {CU42, CU43}을 대각선 방향 인접한 병렬 처리 블록으로 결정할 수도 있다.

[0271] 도 22 (d)는, 예를 들어, 도 22(c)와 동일하게, 코딩 유닛이 하위 뎁스 코딩 유닛 (CU41~CU54)로 분할된 경우를 도시한 것이다. 여기서, 전술한 다양한 병렬 처리 블록의 결정 방식중 적어도 수평, 수직, 대각선 방향중 어느 하나로 인접한 코딩 유닛의 너비 또는 높이 크기에 근거하여, 병렬 처리 블록을 동시에 결정할 수 있다. 예를 들어, 세번째 뎁스에 의해 분할된 {CU52, CU53}을 수평 방향 인접한 병렬 처리 블록으로 결정할 수 있고, 다른 위치의 세번째 뎁스에 의해 분할된 {CU49, CU50}을 수직 방향 인접한 병렬 처리 블록으로 결정할 수 있고, 또한, 다른 위치의 세번째 뎁스에 의해 분할된 {CU41, CU44}를 대각선 방향 인접한 병렬 처리 블록으로 결정할 수도 있다.

[0272] 도 23은 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할에 의해 분할된 병렬 처리 가능한 병렬 코딩 유닛 단위 (PCUR: Parallel Coding Unit processing Region)를 설명하기 위해 도시한 것이다. 부호화 및/또는 복호화 처리 대기 시간 (latency)을 줄이기 위해 기 정의된 특정 영역 내에서는 예측 블록 (인터 예측 부호화 모드 및 인트라 예측 부호화 모드를 통해 생성되는 예측 블록)을 병렬적으로 처리하도록 제한할 수도 있다. 상기 기 정의된 특정 영역을 “병렬 코딩 유닛 단위(Parallel Coding Unit processing Region ,PCUR)”이라고 부른다.

[0273] 즉, NxM PCUR 단위 내에 있는 코딩 영역에 있는 데이터 (예를 들어, 샘플 값,인트라 레퍼런스 샘플, 모션 정보, 레퍼런스 픽쳐 리스트 등)은 서로 참조하지 않도록 제한할 수 있다. 여기서, N과 M은 4, 8, 16, 32 또는 그 이상일 수 있고, N과 M은 동일하거나 상이할 수도 있다. 구체적으로 예를 들어, 도 23 에서는 PCUR의 크기가 8x8인 경우를 도시하였다. 8x8 PCUR 블록 내에 존재하는 코딩 유닛들 {CU10, CU11, CU12} 및 {CU15, CU16} 에서는 데이터(예를 들어, 샘플 값, 인트라 레퍼런스 샘플, 모션 정보, 레퍼런스 픽쳐 리스트 등)을 서로 참조하지 않도록 제한하여 각 CU 간의 독립적인 병렬 처리가 가능하도록 설정할 수 있다.

[0274] 이 경우, 16x16 코딩 유닛 처럼 설정된 PCUR 크기 보다 큰 코딩 유닛이거나, 도 23의 CU13 내지 CU14과 같이 PCUR에 완전히 포함되지 않는 8x16 코딩 유닛은 병렬 처리하지 않고, 순차적으로 혹은 주변 블록에 데이터 디펜던시(dependency)를 가지고 부호화를 수행할 수 있다. 즉 주변 데이터(예를 들어, 샘플 값,인트라 레퍼런스 샘플, 모션 정보, 레퍼런스 픽쳐 리스트 등)를 이용하여 부호화를 수행할 수 있다.

[0275] 단, 병렬 코딩 유닛 단위 PCUR은 정방형 영역일 수도 있고, 비 정방형 영역일 수도 있다. 정방형인지 비 정방형 영역인지를 알려주는 신택스 요소 isRect_PCUR_flag을 시퀀스 파라미터 헤더 또는 픽쳐 파라미터 헤더, 슬라이

스 헤더 등에 시그날링할 수 있다.

[0276] 예를 들어, 병렬 코딩 유닛 단위 PCUR이 8x8인 경우에는 8x8 영역내에 있는 코딩 유닛내에 있는 데이터 (예를 들어, 샘플 값, 인트라 레퍼런스 샘플, 모션 정보, 레퍼런스 픽쳐 리스트 등)은 참조 하지 않도록 정의할 수 있다. 만약 “isRect_PCUR_flag = 1” 이면 병렬 코딩 유닛 단위 PCUR이 정방형임을 나타내고, 값이 “isRect_PCUR_flag = 0” 이면 병렬 코딩 유닛 단위 PCUR이 비 정방형임을 나타내는 것으로 정의할 수 있다.

[0277] 구체적으로, “isRect_PCUR_flag = 1”인 경우에는 병렬 코딩 유닛의 너비 및 높이를 나타내는 신택스 'log2_PCUR_width_minus1'을 더 시그날링 할 수 있다. 이 때 병렬 코딩 유닛의 너비 및 높이는 1 << (log2_PCUR_width_minus1+1) 로 정의할 수 있다. 여기서 << 는 비트 쉬프트 (bit shift) 연산을 나타낸다. 예를 들어, “log2_PCUR_width_minus1 = 1”이면, 최소 병렬 코딩 유닛 단위의 크기는 4x4 단위임을 나타낸다.

[0278] 또한, “isRect_PCUR_flag = 0”인 경우에는 병렬 코딩 유닛의 너비와 높이를 나타내는 신택스를 각각 시그날링 할 수 있다. 또는, isRect_PUCR_flag와 관계없이, 부호화기는 시퀀스, 픽쳐, 또는 슬라이스 별로 최적의 병렬 코딩 유닛의 크기를 결정하고, 이를 부호화하여 시그날링할 수 있다. 또는, 부호화기/복호화기에 기-정의된 고정된 크기의 병렬 코딩 유닛이 이용될 수도 있다.

[0279] 도 24은 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할에 의해 분할된 코딩 블록의 병렬 처리 부호화 방법을 설명하기 위해 도시한 것이다.

[0280] 우선, 멀티 파티셔닝 방식으로 분할된 코딩 블록의 파티션 형태를 결정한다(S2410). 상기 파티션 형태는 전술한 도 11 및 도 12 의 파티션 형태중 어느 하나일 수 있다. 분할된 코딩 블록의 예측 모드를 결정한다(S2420). 예측 모드는 인트라 예측 또는 인터 예측을 포함한다. 단, 인트라 예측 블록에 대해서만, 병렬 부호화 방법을 적용할 수 있다.

[0281] 상기 결정된 파티션 형태 및 예측 모드에 따라, 병렬 처리 가능한 블록을 결정한다 (S2430). 병렬 처리 블록 결정 기준은, 전술한 바와 같이 다양한 방법에 의해 결정될 수 있다. 예를 들어, 코딩 블록 샘플의 너비 및 높이 크기 (임계치 이하), 인접한 병렬 코딩 블록과의 위치 (수평, 수직, 대각선 인접) 에 의해 결정될 수 있다. 또한, 전술한 병렬 코딩 유닛 단위 PCUR 설정에 의해 병렬 코딩 블록이 결정될 수 있다.

[0282] 만약, 상기 단계 S2430에 의해, 병렬 처리 블록으로 결정되지 않으면, 각 코딩 블록별 참조 샘플을 결정하고 (S2460), 코딩 블록 단위로 부호화를 수행한다(S2470). 반면, 상기 단계 S2430에 의해, 병렬 처리 블록으로 결정되면, 병렬 처리 블록들을 독립적으로 동시 처리하게 된다. 이를 위해 병렬 블록별 참조 샘플을 결정하고 (S2440), 병렬 처리 블록들을 동시에 부호화하게 된다(S2450). 상기 단계 S2440의 병렬 블록의 참조 샘플 결정은 도 26~29에서 상세히 후술할 예정이다.

[0283] 도 25는 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할에 의해 분할된 코딩 블록의 병렬 처리 복호화 방법을 설명하기 위해 도시한 것이다.

[0284] 우선, 입력되는 인코딩된 비디오 비트스트림으로부터, 부호화 정보를 파싱한다(S2510). 상기 파싱된 부호화 정보로부터, 멀티 파티셔닝 방식으로 분할된 코딩 블록의 파티션 형태 및 각 분할된 코딩 블록의 예측 모드를 확인한다(S2520). 또한, 파싱된 부호화 정보로부터, 전술한 병렬 코딩 유닛 단위 PCUR 설정여부 및 설정된 PCUR 크기를 확인할 수 있다(S2520). 관련하여, 상기 파티션 형태는 전술한 도 11 및 도 12 의 파티션 형태중 어느 하나일 수 있다. 또한, 코딩 블록의 예측 모드는 인트라 예측 또는 인터 예측을 포함한다. 단, 인트라 예측 블록에 대해서만, 병렬 복호화 방법을 적용할 수 있다.

[0285] 상기 결정된 파티션 형태 및 예측 모드에 따라, 병렬 처리 가능한 블록을 결정한다 (S2530). 병렬 처리 블록 결정 기준은, 전술한 바와 같이 다양한 방법에 의해 결정될 수 있다. 예를 들어, 코딩 블록 샘플의 너비 및 높이 크기 (임계치 이하), 인접한 병렬 코딩 블록과의 위치 (수평, 수직, 대각선 인접) 에 의해 결정될 수 있다. 또한, 전술한 병렬 코딩 유닛 단위 PCUR 설정에 의해 병렬 코딩 블록이 결정될 수 있다.

[0286] 만약, 상기 단계 S2530에 의해, 병렬 처리 블록으로 결정되지 않으면, 각 코딩 블록별 참조 샘플을 결정하고 (S2560), 코딩 블록 단위로 복호화를 수행한다(S2570). 반면, 상기 단계 S2530에 의해, 병렬 처리 블록으로 결정되면, 병렬 처리 블록들을 독립적으로 동시 처리하게 된다. 이를 위해 병렬 블록별 참조 샘플을 결정하고 (S2540), 병렬 처리 블록들을 동시에 복호화하게 된다(S2550). 이하 상기 단계 S2540의 병렬 블록의 참조 샘플 결정은 도 26~29 에서 상세히 후술할 것이다.

[0287] 도 26 및 도 27은, 전술한 도20에 도시된 상기 제1타입 코딩 블록의 병렬 처리를 위해 참조 샘플을 획득하는 방

법을 설명하기 위해 도시한 것이다. 우선, 도 26 및 도 27에 도시된 이웃하는 코딩 유닛 CU5 및 CU6은 전술한 병렬 처리 블록 기준에 해당되어 독립적인 병렬 처리 복호화 및/또는 부호화가 수행되는 코딩 유닛을 의미한다. 예를 들어, 2x8 크기의 CU5 및 CU6은 전술한 다양한 병렬 처리 기준 중, 수평방향으로 인접한 동일 크기의 코딩 유닛에 해당되고, 코딩 유닛의 너비(w)가 4 이하 이고, 너비(w)와 높이(h) 비 즉, w/h값이 특정 임계값(예, 1/2)보다 작은지 여부에 기초하여 병렬 처리 블록으로 결정될 수 있다. 단, 전술한 바와 같이 상기 병렬 처리 블록 결정 기준은 부호화 성능 및 효율을 고려하여 선택적으로 설정 가능하다.

[0288] 도 26과 같이, 병렬 처리 대상인 2x8 코딩 유닛 CU5 주변에 이미 복원된 영역이 존재하는 경우에는 복원된 영역의 샘플 값들을 활용하여 참조 샘플을 구성할 수 있다. 또한, 만약 CU5 주변에 일부 복원되지 않은 영역의 샘플 값들이 존재하는 경우에는, 이미 복원된 주변 샘플 값들을 이용하여 참조 샘플을 구성할 수도 있다.

[0289] 하지만 도27과 같이 또 다른 병렬 처리 대상인 코딩 유닛 CU6의 좌측으로는 복원된 영역이 존재 하지 않을 시에는, CU6의 좌측 참조 샘플 R(-1,i)을, 이웃 코딩 유닛 CU5의 참조 샘플 값을 활용하거나 또는 기준점 샘플들간의 샘플 값 차이를 나타내는 오프셋을 이용하여 유도할 수 있다.

[0290] 구체적으로, 코딩 유닛 CU5와 CU6의 각 좌상단 또는 상단 샘플을 기준점 샘플로 설정하고, 이들 기준점 샘플 간의 차이 값을 오프셋으로 설정할 수 있다. 예를 들어, 도 27에서, CU5의 기준점은 P(0,-1) (2702), CU6의 기준점은 P(2,-1) (2704)로 설정하고, 상기 두 코딩 유닛 (CU5, CU6)의 기준점 샘플들간의 샘플 값 차이를 오프셋으로 설정할 수 있다. 즉, 다음 수학식 (1)과 같이, 오프셋(f)를 설정할 수 있다.

수학식 1

[0292] $$f = P(0, -1) - P(w, -1)$$

[0294] 수학식 (1) 에서 w는 코딩 유닛의 너비를 나타내고, f는 대응 CU의 기준점 샘플 값에서 대상 CU의 기준점 샘플 값의 차이를 나타낸다.

[0295] 단, 상기 기준점 위치는 다양한 방식에 의해 결정하는 것이 가능하며, 예를 들어, CU5의 기준점은 P(-1,-1) (2701)로, CU6의 기준점은 P(1,-1) (2703)로 설정하고 오프셋(f)을 구하는 것도 가능하다. 또한, CU5의 기준점은 P(-1,-1) (2701) 및 P(0,-1) (2702)의 평균값으로, CU6의 기준점은 P(1,-1) (2703) 및 P(2,-1) (2704)의 평균값으로 설정하고 오프셋(f)을 구하는 것도 가능하다.

[0296] 따라서, CU5의 참조 샘플들중 복원되지 않은 좌측 참조 샘플들 R(-1,i)은 CU6의 좌측 참조 샘플에서 상기 오프셋(f)을 차분한 값으로 유도할 수도 있으며, 다음과 같이 수학식 (2)로 표현 할 수 있다.

수학식 2

[0298] $$R(-1, i) = P(-1, i) - f$$

[0300] 도 28 및 도 29는 또 다른 실시예로서, 상기 제2타입 코딩 블록의 병렬 처리를 위해 참조 샘플을 획득하는 방법을 설명하기 위해 도시한 것이다. 우선, 도 28 및 도 29에 도시된 이웃하는 코딩 유닛 CU10 및 CU11은 전술한 병렬 처리 블록 기준에 해당되어 독립적인 병렬 처리 복호화 및/또는 부호화가 수행되는 코딩 유닛을 의미한다. 예를 들어, 8x2 크기의 CU10 및 CU11은 전술한 다양한 병렬 처리 기준 중, 수직방향으로 인접한 동일 크기의 코딩 유닛에 해당되고, 코딩 유닛의 높이(h)가 4 이하 이고, 너비(w)와 높이(h) 비 즉, h/w값이 특정 임계값(예,

1/2)보다 작은지 여부에 기초하여 병렬 처리 블록으로 결정될 수 있다. 단, 전술한 바와 같이 상기 병렬 처리 블록 결정 기준은 부호화 성능 및 효율을 고려하여 선택적으로 설정 가능하다.

[0301] 도 28과 같이, 병렬 처리 대상인 8x2 코딩 유닛 CU10 주변에 이미 복원된 영역이 존재하는 경우에는 복원된 영역의 샘플 값들을 활용하여 참조 샘플을 구성할 수 있다. 또한, 만약 CU10 주변에 일부 복원되지 않은 영역의 샘플 값들이 존재하는 경우에는, 이미 복원된 주변 샘플 값들을 이용하여 참조 샘플을 구성할 수도 있다.

[0302] 하지만 도29와 같이 또 다른 병렬 처리 대상인 코딩 유닛 CU11의 상단으로는 복원된 영역이 존재 하지 않을 시에는, CU11의 상단 참조 샘플 R(i, -1)을, 이웃 코딩 유닛 CU10의 참조 샘플 값을 활용하거나 또는 기준점 샘플들간의 샘플 값 차이를 나타내는 오프셋을 이용하여 유도할 수 있다.

[0303] 구체적으로, 코딩 유닛 CU10와 CU11의 각 좌상단 또는 좌측 샘플을 기준점 샘플로 설정하고, 이들 기준점 샘플간의 차이 값을 오프셋으로 설정할 수 있다. 예를 들어, 도 29에서, CU10의 기준점은 P(-1,0) (2902), CU11의 기준점은 P(-1,2) (2904)로 설정하고, 상기 두 코딩 유닛 (CU10, CU11)의 기준점 샘플들간의 샘플 값 차이를 오프셋으로 설정할 수 있다. 즉, 다음 수학식 (3)과 같이, 오프셋(f)를 설정할 수 있다.

수학식 3

[0305] $$f = P(-1, 0) - P(-1, h)$$

[0307] 수학식 (3) 에서 h는 코딩 유닛의 높이를 나타내고, f는 대응 CU의 기준점 샘플 값에서 대상 CU의 기준점 샘플 값의 차이를 나타낸다.

[0308] 단, 상기 기준점 위치는 다양한 방식에 의해 결정하는 것이 가능하며, 예를 들어, CU10의 기준점은 P(-1,-1) (2901)로, CU11의 기준점은 P(-1,1) (2903)로 설정하고 오프셋(f)을 구하는 것도 가능하다. 또한, CU10의 기준점은 P(-1,-1) (2901) 및 P(-1,0) (2902)의 평균값으로, CU11의 기준점은 P(-1,1) (2903) 및 P(-1,2) (2904)의 평균값으로 설정하고 오프셋(f)을 구하는 것도 가능하다.

[0309] 따라서, CU11의 참조 샘플들중 복원되지 않은 상단 참조 샘플들 R(i,-1)은 CU10의 상단 참조 샘플에서 상기 오프셋(f)을 차분한 값으로 유도할 수도 있으며, 다음과 같이 수학식 (4)로 표현 할 수 있다.

수학식 4

[0311] $$R(i, -1) = P(i, -1) - f$$

[0313] 도 30은 본 발명이 적용되는 또 다른 실시예로서, 멀티 트리 분할이 분할된 병렬 처리 코딩 블록이 적용되는 네트워크 추상화 계층 (NAL)에 포함되는 신택스 요소(syntax element)를 예를 들어 도시한 것이다.

[0314] 본 발명이 적용되는 NAL 유닛은, 예를 들어, 비디오 파라미터 셋 (VPS), 시퀀스 파라미터 셋(SPS), 픽쳐 파라미터 셋(PPS) 및 적어도 하나 이상의 슬라이스 셋(Slice)을 포함할 수 있다.

[0315] 예를 들어, 도 30에서는 픽쳐 파라미터 셋(PPS)에 포함된 포함된 신택스 요소를 도시하였으나, 시퀀스 파라미터 셋(SPS) 또는 슬라이스 셋(Slice)에 해당 신택스 요소를 포함하는 것도 가능하다. 또한, 신택스 요소별로 시퀀스 단위 또는 픽쳐 단위에 공통적으로 적용될 신택스 요소는 시퀀스 파라미터 셋(SPS) 또는 픽쳐 파라미터 셋(PPS)에 포함되도록 할 수 있다. 반면, 해당 슬라이스에만 적용되는 신택스 요소는 슬라이스 셋(Slice)에 포함되는 것이 바람직하다. 따라서, 이는 부호화 성능 및 효율을 고려하여 선택이 가능하다.

[0316] 신택스 요소 'isRect_PCUR_flag'는 병렬 코딩 유닛 단위 PCUR이 정방형 영역 인지 비 정방형 영역인지를 나타낸다. 예를 들어, “isRect_PCUR_flag = 1” 이면 병렬 코딩 유닛 단위 PCUR이 정방형임을 나타내고, “isRect_PCUR_flag = 0” 이면 병렬 코딩 유닛 단위 PCUR이 비 정방형임을 나타내도록 정의할 수 있다.

[0317] 또한, 신택스 요소 'log2_PCUR_width_minus1'는 병렬 코딩 유닛 단위 PCUR이 정방형인 경우, PCUR의 크기를 나타낸다. 예를 들어, “log2_PCUR_width_minus1 = 1”이면, 최소 병렬 코딩 유닛 단위의 크기는 4x4 단위임을 나타내도록 정의할 수 있다.

[0319] 상술한 실시예는 일련의 단계 또는 순서도를 기초로 설명되고 있으나, 이는 발명의 시계열적 순서를 한정한 것은 아니며, 필요에 따라 동시에 수행되거나 다른 순서로 수행될 수 있다. 또한, 상술한 실시예에서 블록도를 구성하는 구성요소(예를 들어, 유닛, 모듈 등) 각각은 하드웨어 장치 또는 소프트웨어로 구현될 수도 있고, 복수의 구성요소가 결합하여 하나의 하드웨어 장치 또는 소프트웨어로 구현될 수도 있다. 상술한 실시예는 다양한 컴퓨터 구성요소를 통하여 수행될 수 있는 프로그램 명령어의 형태로 구현되어 컴퓨터 판독 가능한 기록 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능한 기록 매체는 프로그램 명령어, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 컴퓨터 판독 가능한 기록 매체의 예에는, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체, CD-ROM, DVD와 같은 광기록 매체, 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 ROM, RAM, 플래시 메모리 등과 같은 프로그램 명령어를 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 상기 하드웨어 장치는 본 발명에 따른 처리를 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있다.

부호의 설명

[0321] 100 : 부호화기 110 : 픽쳐분할부

120, 230 : 인터 예측부 125, 235 : 인트라 예측부

130 : 변환부 135 : 양자화부

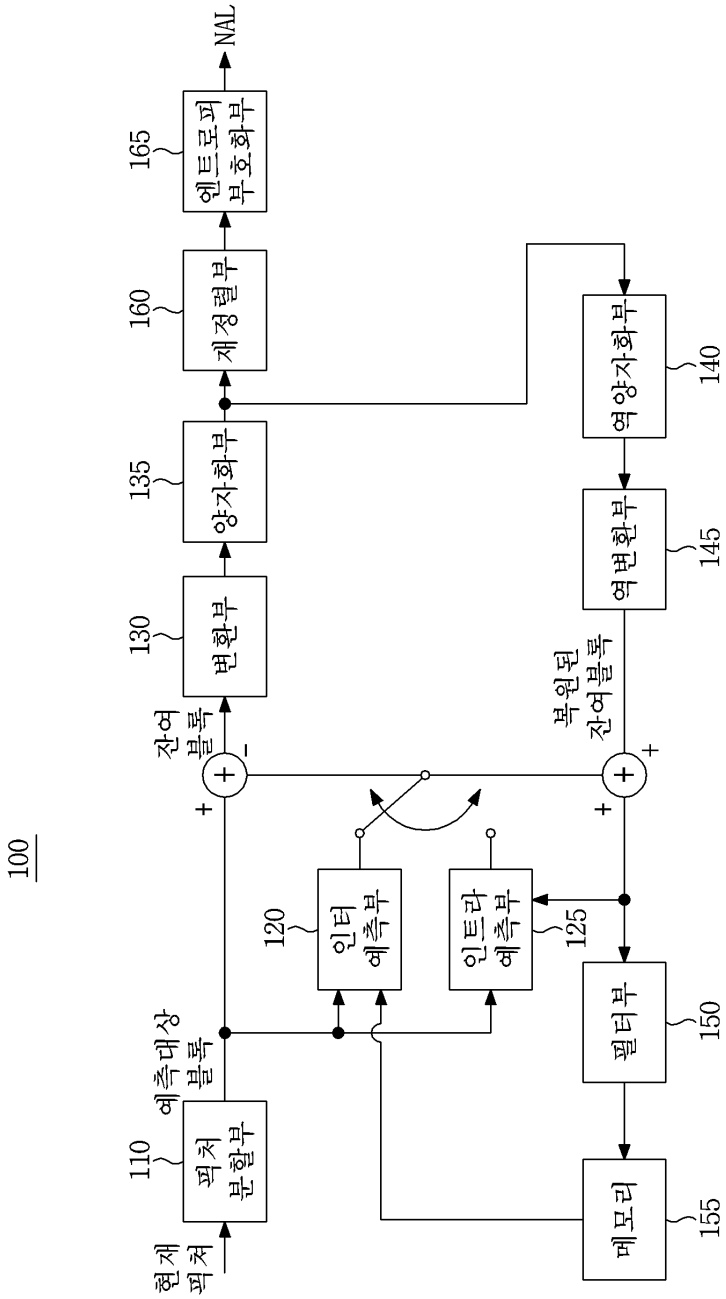
200 : 복호화기 210 : 엔트로피 복호화부

215 : 재정렬부 220 : 역양자화부

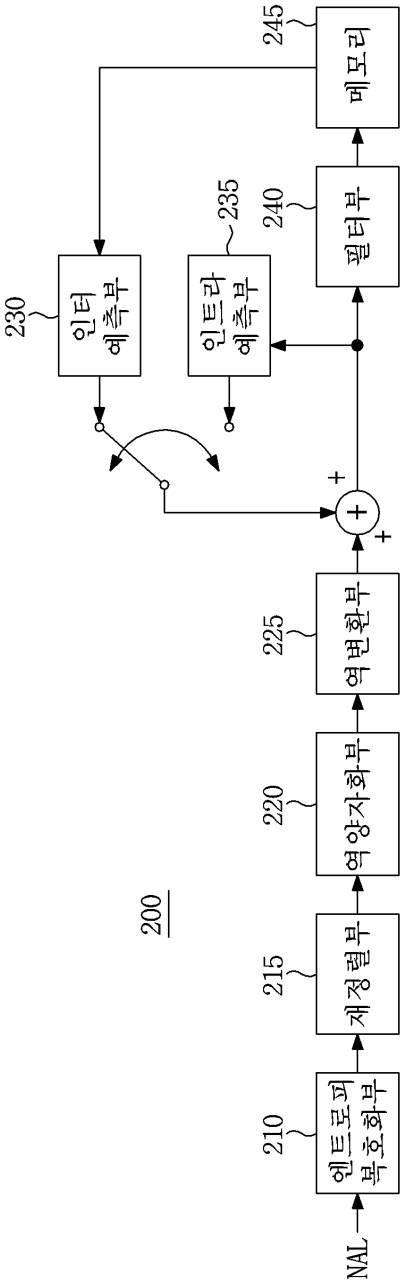
225 : 역변환부

도면

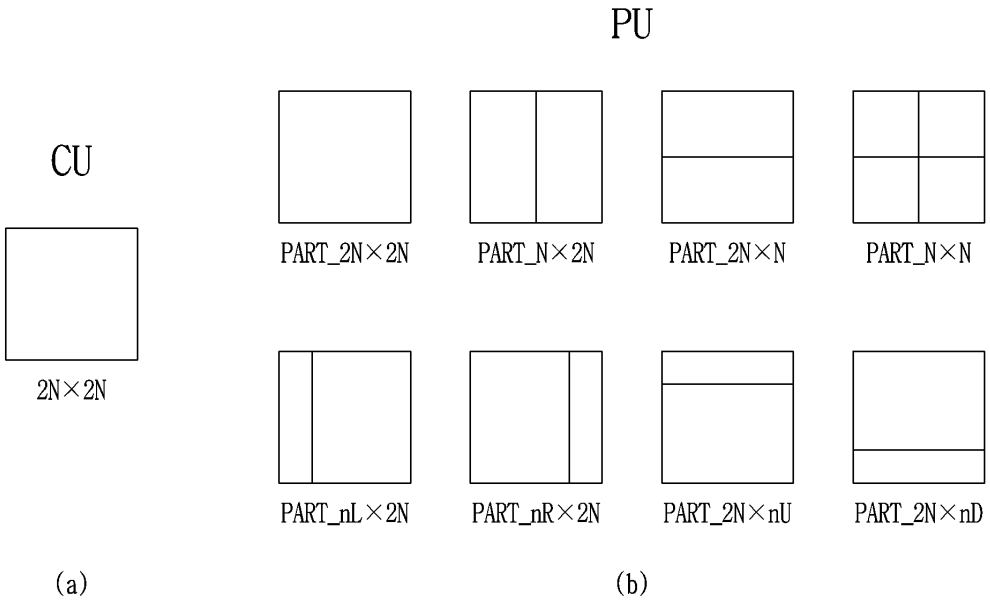
도면1


100
현재 픽처
110
픽처 분할부
예측대상 블록
120
인터 예측부
125
인트라 예측부
잔여 블록
130
변환부
135
양자화부
160
재정렬부
165
엔트로피 부호화부
NAL
복원된 잔여블록
140
역양자화부
145
역변환부
150
필터부
155
메모리

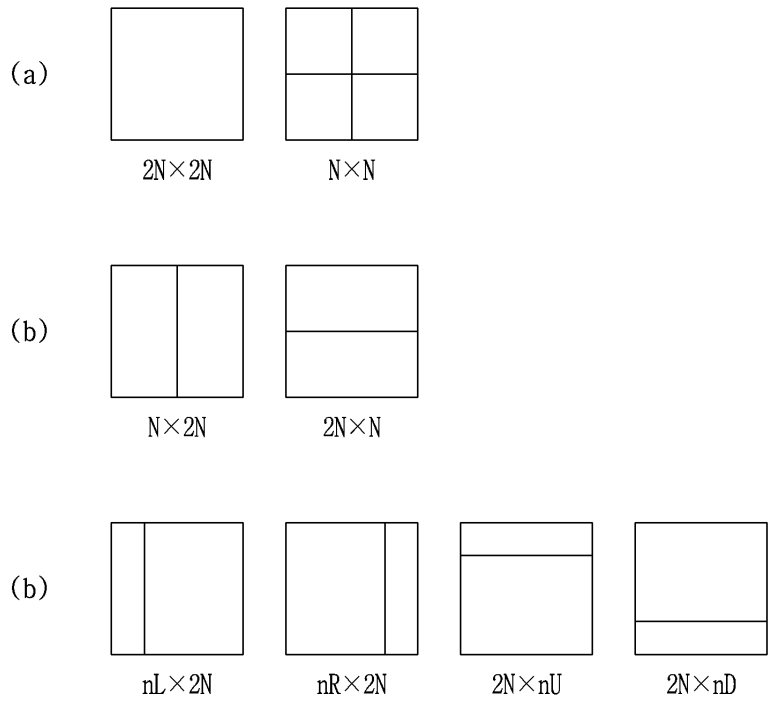
도면2


200
NAL
엔트로피 복호화부
210
재정렬부
215
역양자화부
220
역변환부
225
+
+
인터 예측부
230
인트라 예측부
235
필터부
240
메모리
245

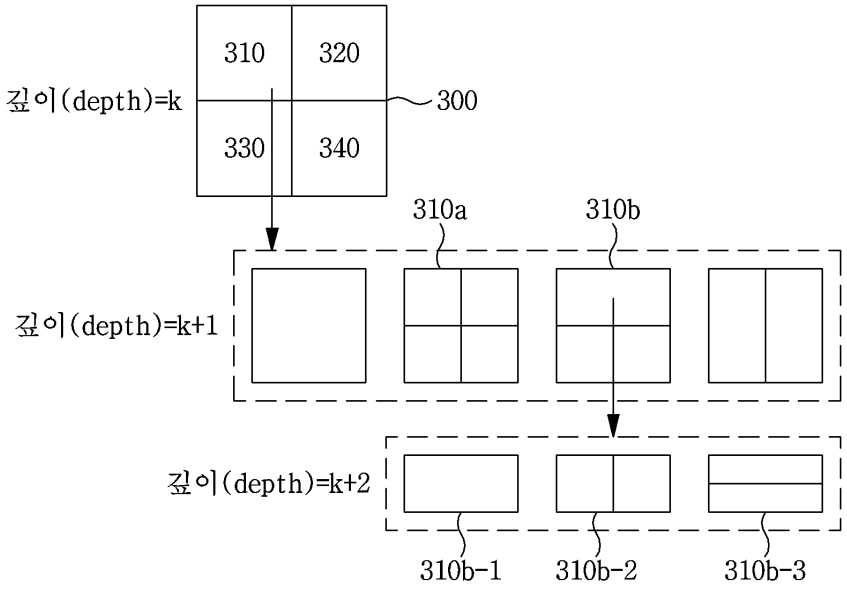
도면3


PU
CU
2N×2N
PART_2N×2N
PART_N×2N
PART_2N×N
PART_N×N
PART_nL×2N
PART_nR×2N
PART_2N×nU
PART_2N×nD
(a)
(b)

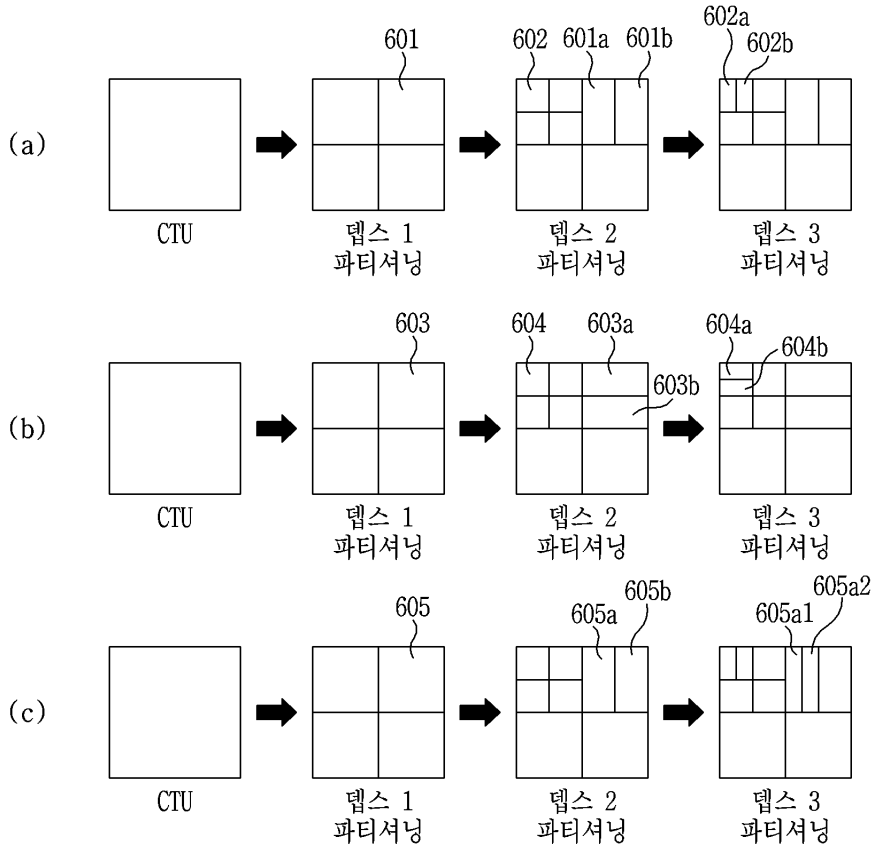
도면4


(a)
2N×2N
N×N
(b)
N×2N
2N×N
(b)
nL×2N
nR×2N
2N×nU
2N×nD

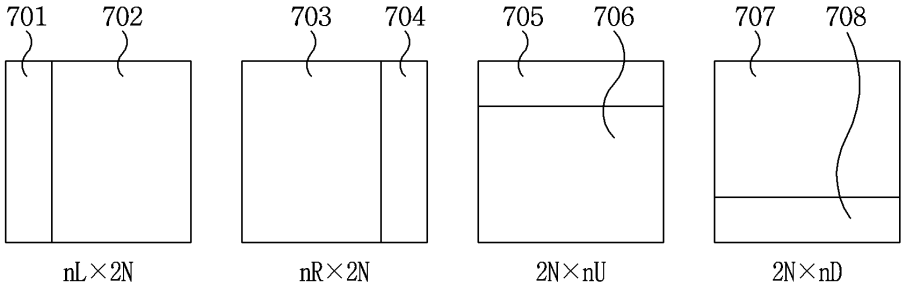
도면5


310
320
330
340
300
깊이(depth)=k
310a
310b
깊이(depth)=k+1
깊이(depth)=k+2
310b-1
310b-2
310b-3

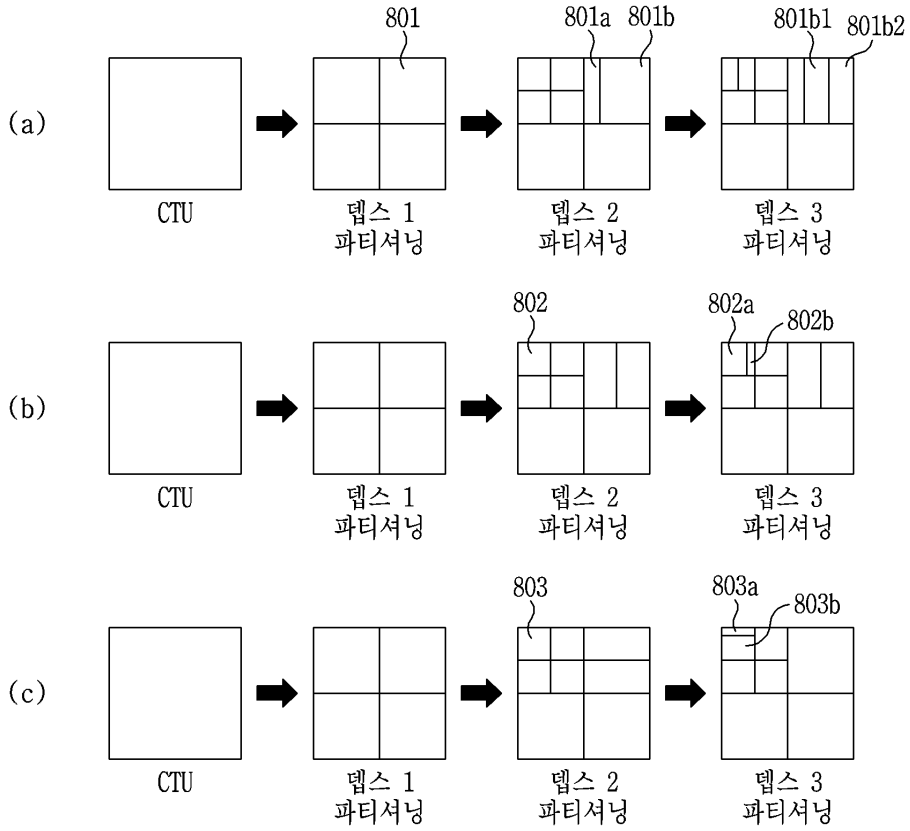
도면6


(a)
601
602
601a
601b
602a
602b
CTU
뎁스 1
파티셔닝
뎁스 2
파티셔닝
뎁스 3
파티셔닝
(b)
603
604
603a
604a
604b
603b
CTU
뎁스 1
파티셔닝
뎁스 2
파티셔닝
뎁스 3
파티셔닝
(c)
605
605a
605b
605a1
605a2
CTU
뎁스 1
파티셔닝
뎁스 2
파티셔닝
뎁스 3
파티셔닝

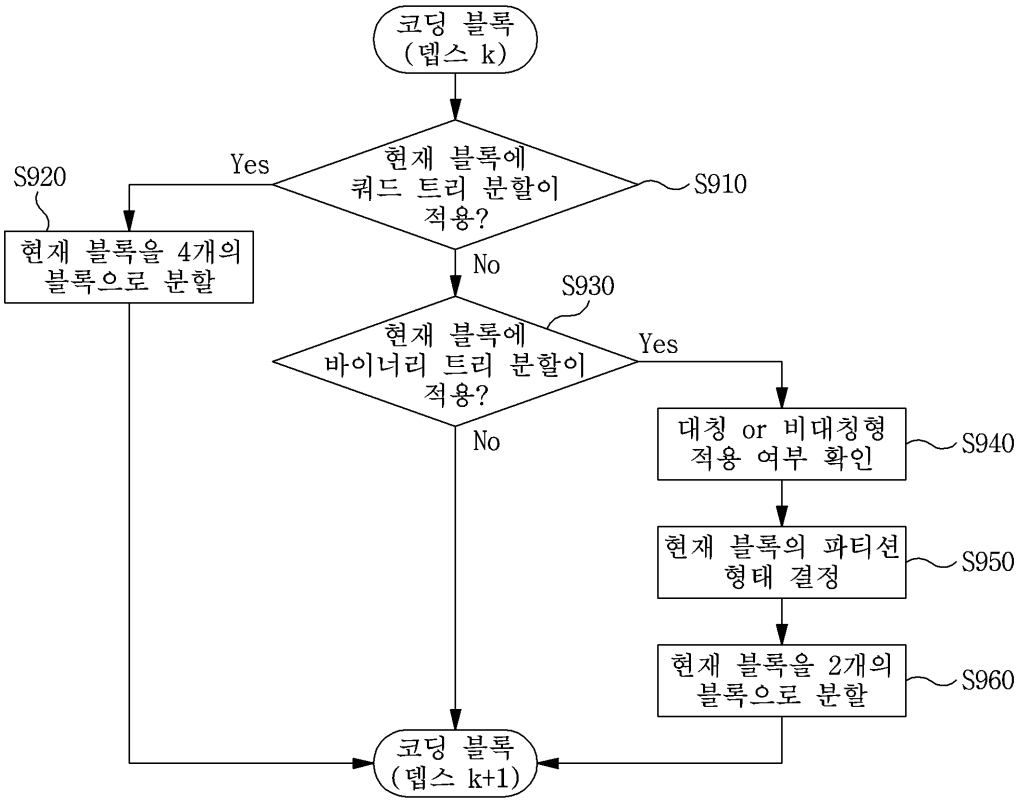
도면7


701
702
703
704
705
706
707
708
nL×2N
nR×2N
2N×nU
2N×nD

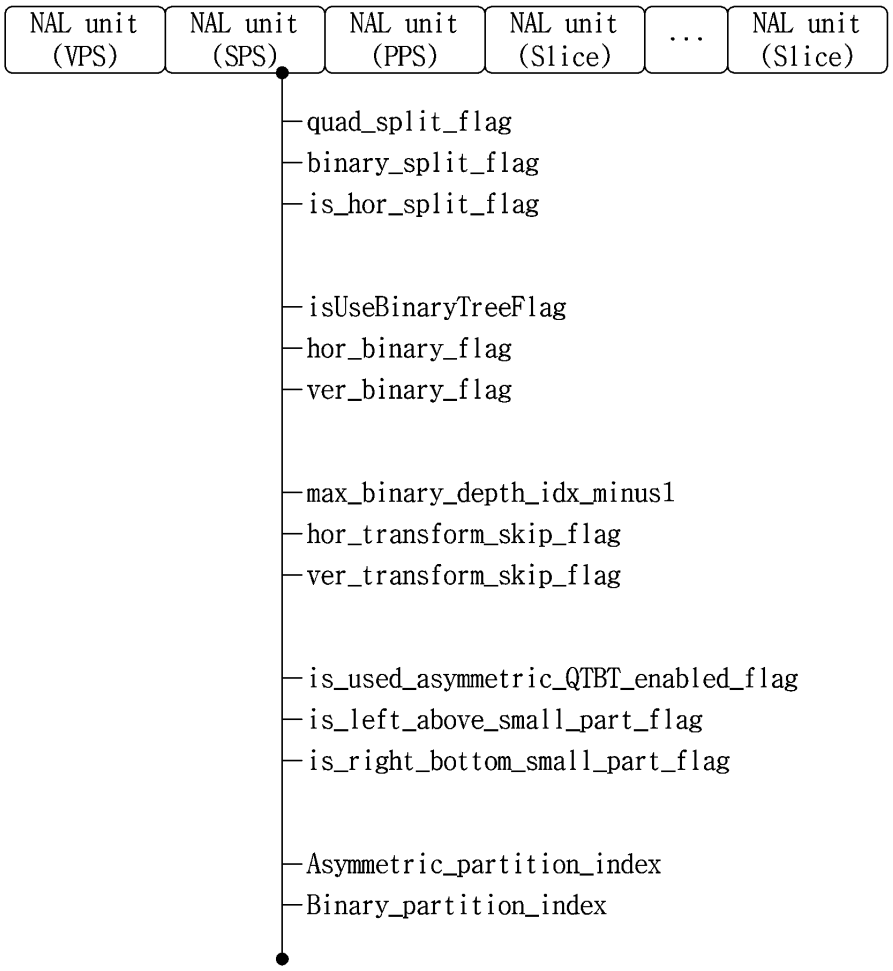
도면8


(a)
801
801a
801b
801b1
801b2
CTU
뎁스 1
파티셔닝
뎁스 2
파티셔닝
뎁스 3
파티셔닝
(b)
802
802a
802b
CTU
뎁스 1
파티셔닝
뎁스 2
파티셔닝
뎁스 3
파티셔닝
(c)
803
803a
803b
CTU
뎁스 1
파티셔닝
뎁스 2
파티셔닝
뎁스 3
파티셔닝

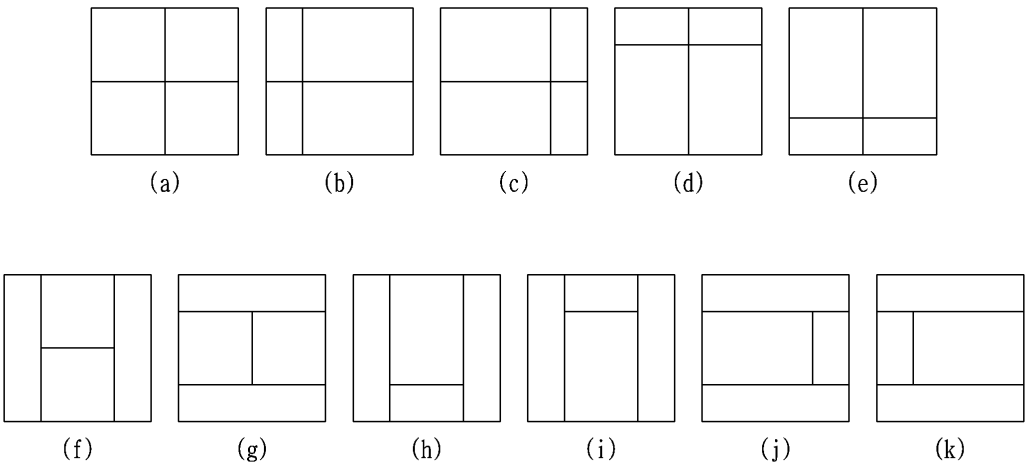
도면9


코딩 블록
(뎁스 k)
현재 블록에
쿼드 트리 분할이
적용?
S910
Yes
S920
현재 블록을 4개의
블록으로 분할
No
S930
현재 블록에
바이너리 트리 분할이
적용?
Yes
No
대칭 or 비대칭형
적용 여부 확인
S940
현재 블록의 파티션
형태 결정
S950
현재 블록을 2개의
블록으로 분할
S960
코딩 블록
(뎁스 k+1)

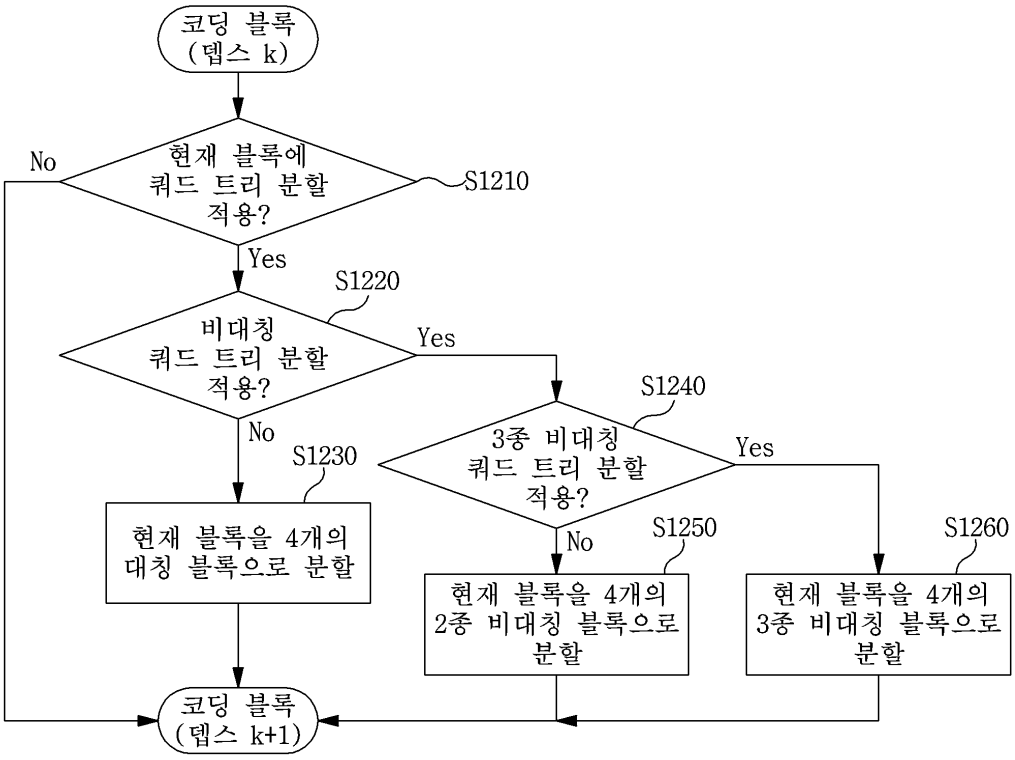
도면10


NAL unit (VPS)
NAL unit (SPS)
NAL unit (PPS)
NAL unit (Slice)
...
NAL unit (Slice)
quad_split_flag
binary_split_flag
is_hor_split_flag
isUseBinaryTreeFlag
hor_binary_flag
ver_binary_flag
max_binary_depth_idx_minus1
hor_transform_skip_flag
ver_transform_skip_flag
is_used_asymmetric_QTBT_enabled_flag
is_left_above_small_part_flag
is_right_bottom_small_part_flag
Asymmetric_partition_index
Binary_partition_index

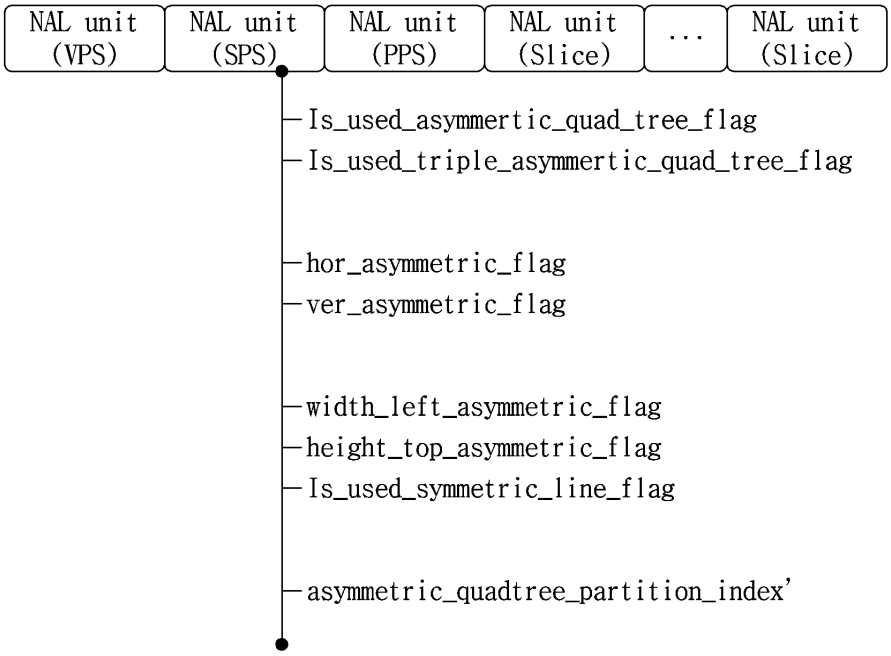
도면11


(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
(j)
(k)

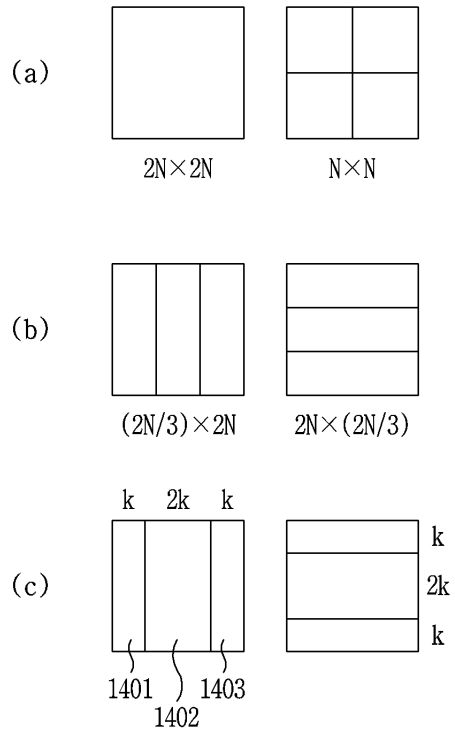
도면12


코딩 블록
(뎁스 k)
현재 블록에
쿼드 트리 분할
적용?
S1210
No
Yes
S1220
비대칭
쿼드 트리 분할
적용?
Yes
No
S1230
현재 블록을 4개의
대칭 블록으로 분할
S1240
3종 비대칭
쿼드 트리 분할
적용?
Yes
No
S1250
현재 블록을 4개의
2종 비대칭 블록으로
분할
S1260
현재 블록을 4개의
3종 비대칭 블록으로
분할
코딩 블록
(뎁스 k+1)

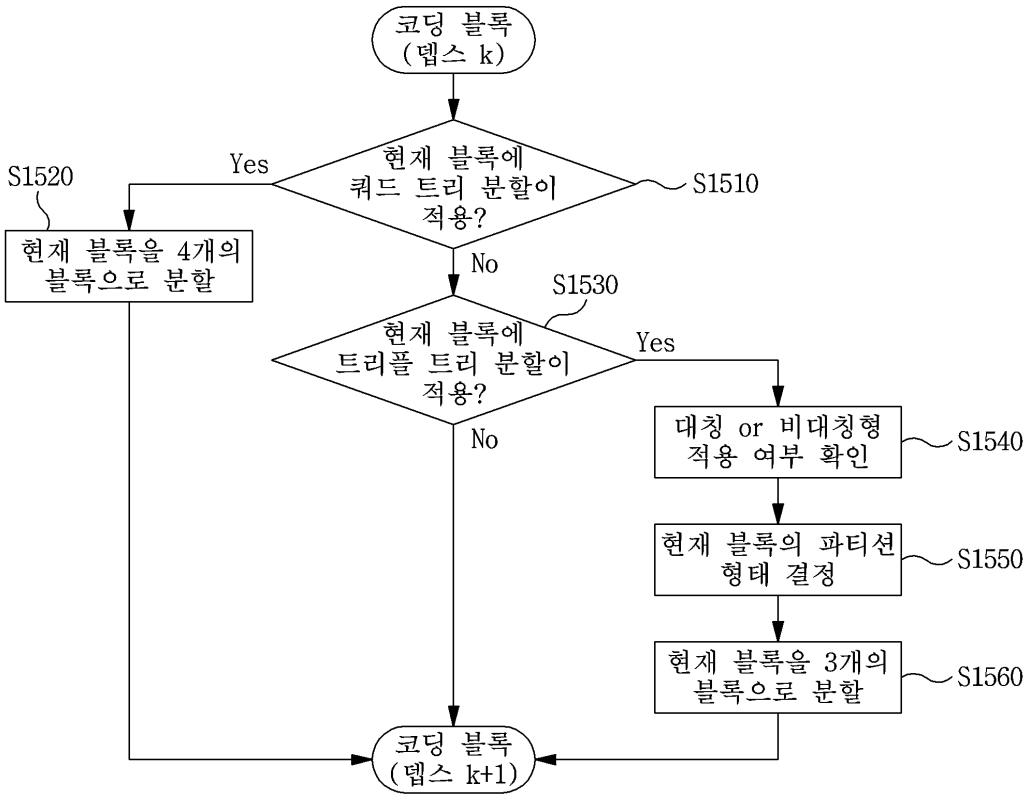
도면13


NAL unit
(VPS)
NAL unit
(SPS)
NAL unit
(PPS)
NAL unit
(Slice)
...
NAL unit
(Slice)
Is_used_asymmertic_quad_tree_flag
Is_used_triple_asymmertic_quad_tree_flag
hor_asymmetric_flag
ver_asymmetric_flag
width_left_asymmetric_flag
height_top_asymmetric_flag
Is_used_symmetric_line_flag
asymmetric_quadtree_partition_index'

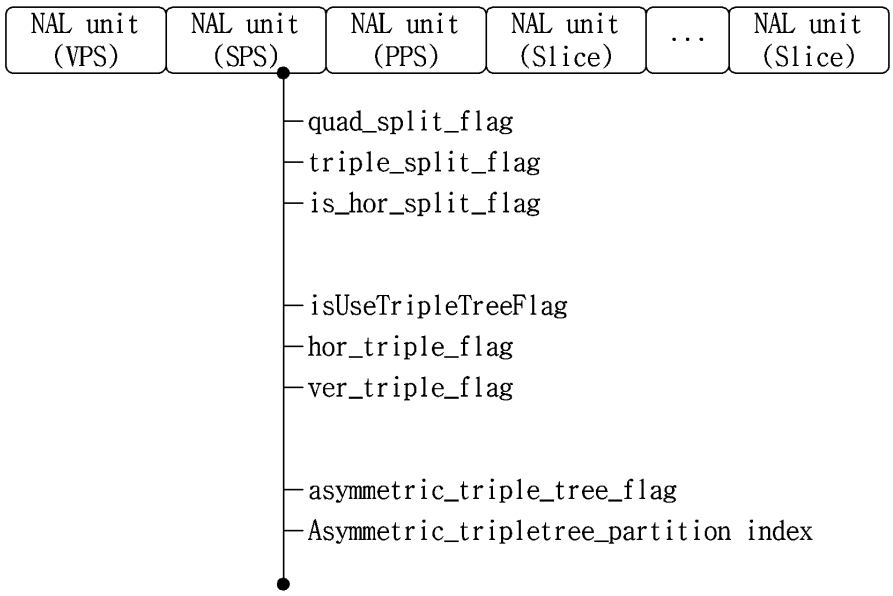
도면14


(a)
2N×2N
N×N
(b)
(2N/3)×2N
2N×(2N/3)
(c)
k
2k
k
1401
1402
1403

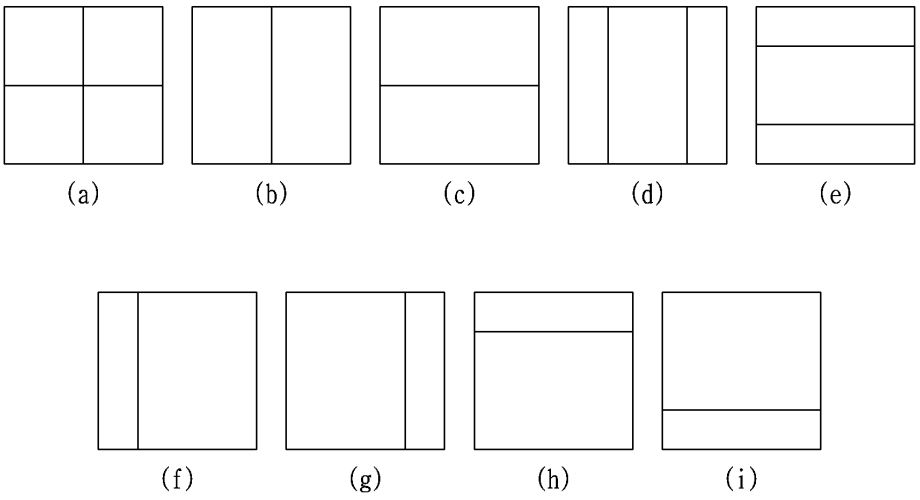
도면15


코딩 블록
(뎁스 k)
현재 블록에
쿼드 트리 분할이
적용?
S1510
Yes
S1520
현재 블록을 4개의
블록으로 분할
No
S1530
현재 블록에
트리플 트리 분할이
적용?
Yes
No
대칭 or 비대칭형
적용 여부 확인
S1540
현재 블록의 파티션
형태 결정
S1550
현재 블록을 3개의
블록으로 분할
S1560
코딩 블록
(뎁스 k+1)

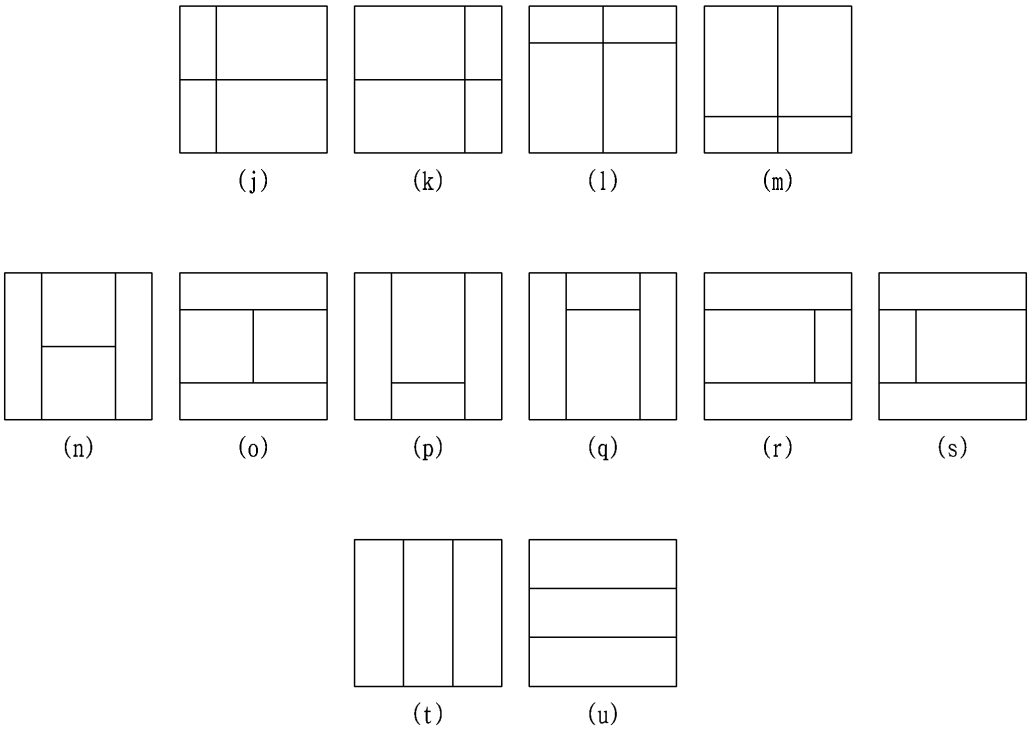
도면16


NAL unit (VPS)
NAL unit (SPS)
NAL unit (PPS)
NAL unit (Slice)
...
NAL unit (Slice)
quad_split_flag
triple_split_flag
is_hor_split_flag
isUseTripleTreeFlag
hor_triple_flag
ver_triple_flag
asymmetric_triple_tree_flag
Asymmetric_tripletree_partition index

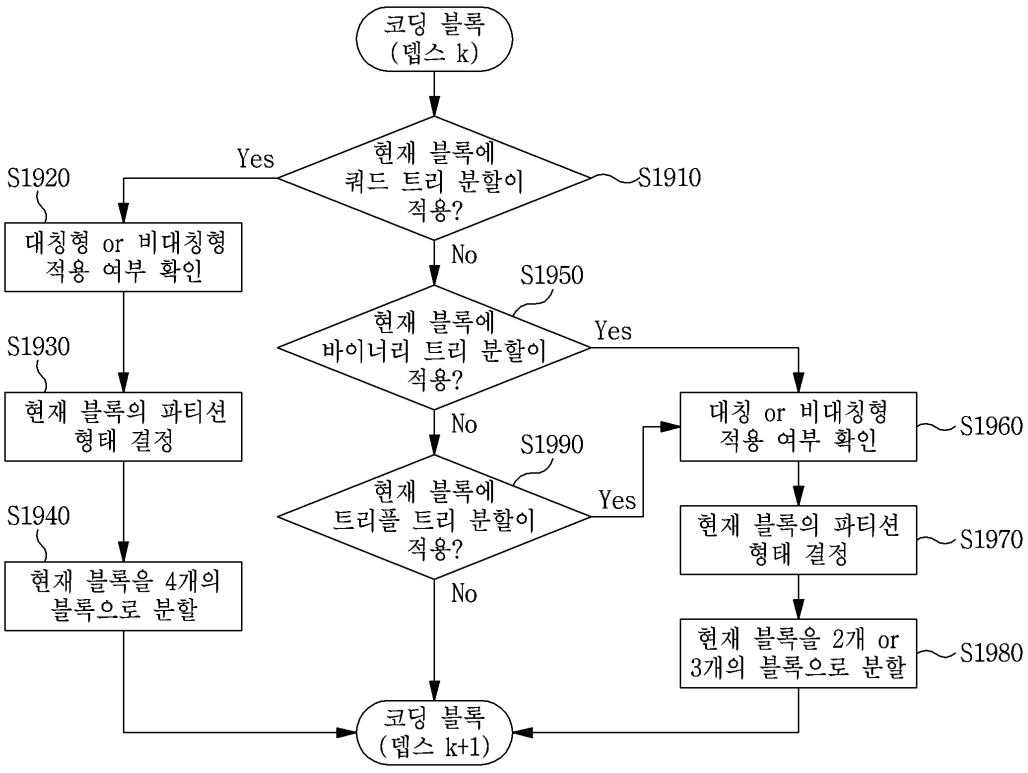
도면17


(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)

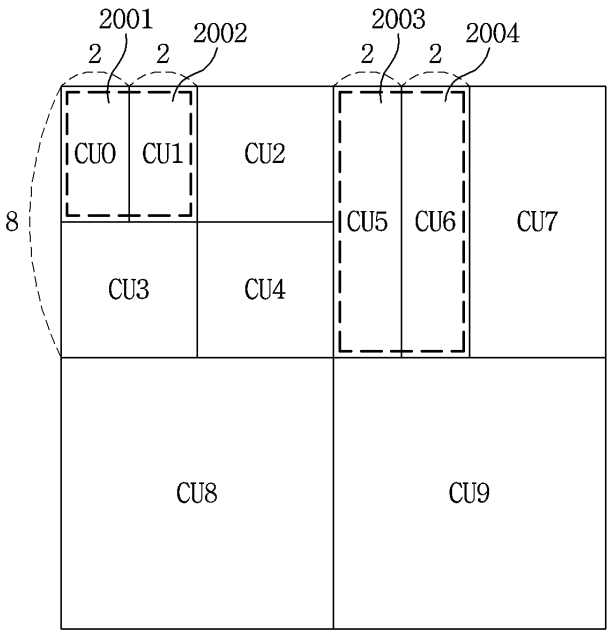
도면18


(j)
(k)
(l)
(m)
(n)
(o)
(p)
(q)
(r)
(s)
(t)
(u)

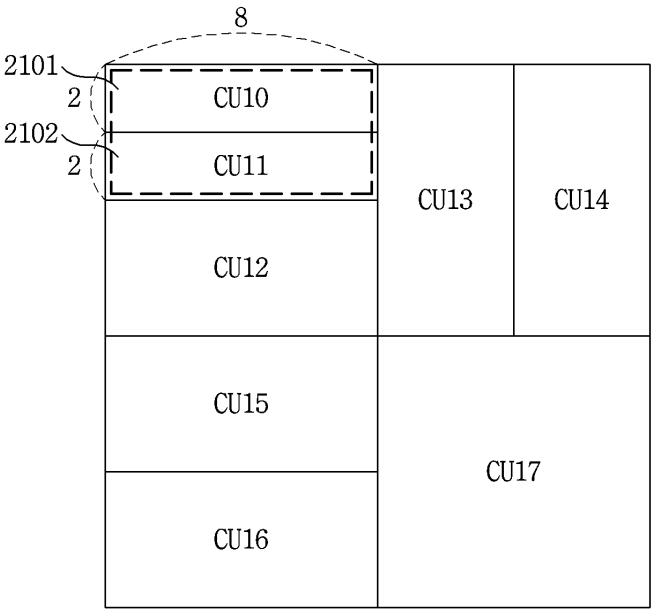
도면19


코딩 블록
(뎁스 k)
현재 블록에
쿼드 트리 분할이
적용?
S1910
Yes
No
S1920
대칭형 or 비대칭형
적용 여부 확인
S1930
현재 블록의 파티션
형태 결정
S1940
현재 블록을 4개의
블록으로 분할
S1950
현재 블록에
바이너리 트리 분할이
적용?
Yes
No
S1960
대칭 or 비대칭형
적용 여부 확인
S1970
현재 블록의 파티션
형태 결정
S1980
현재 블록을 2개 or
3개의 블록으로 분할
S1990
현재 블록에
트리플 트리 분할이
적용?
Yes
No
코딩 블록
(뎁스 k+1)

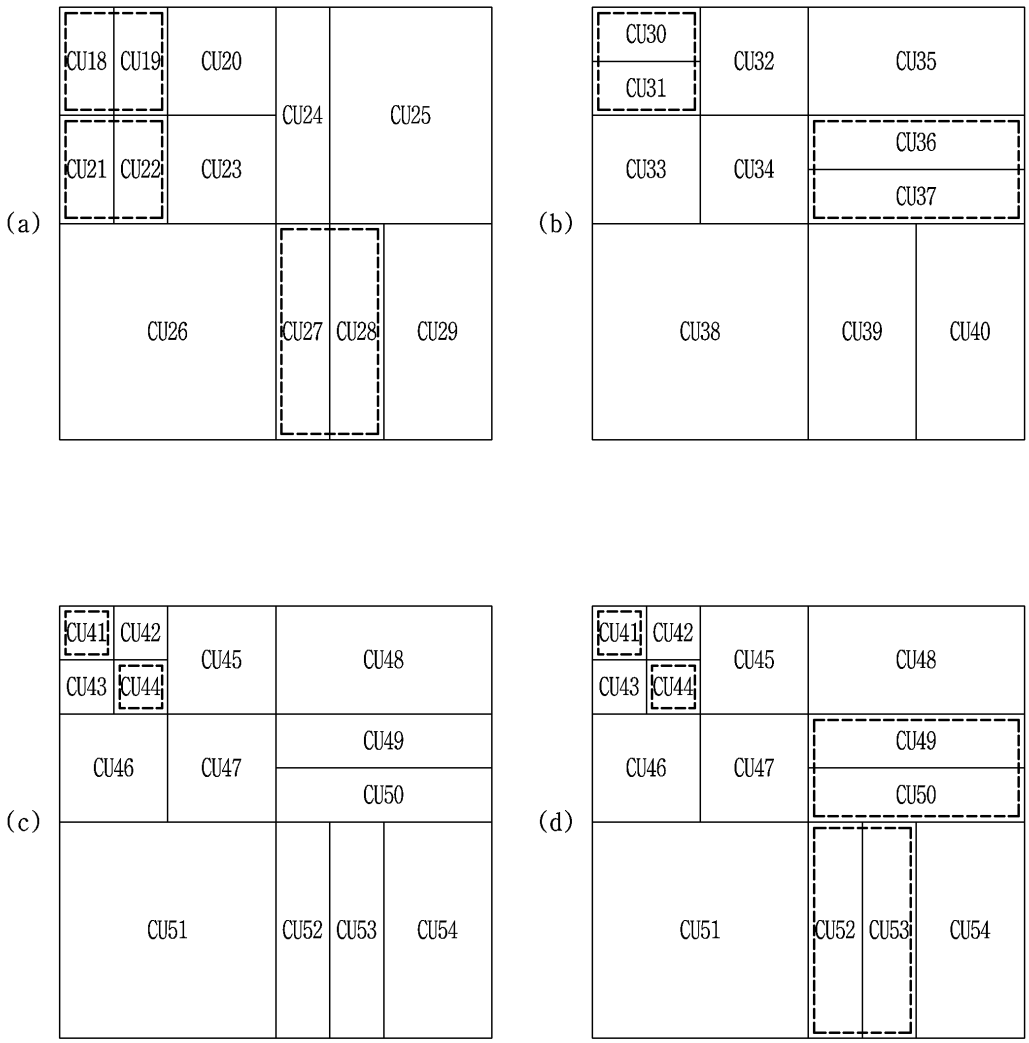
도면20


2001
2002
2003
2004
2
2
2
2
8
CU0
CU1
CU2
CU3
CU4
CU5
CU6
CU7
CU8
CU9

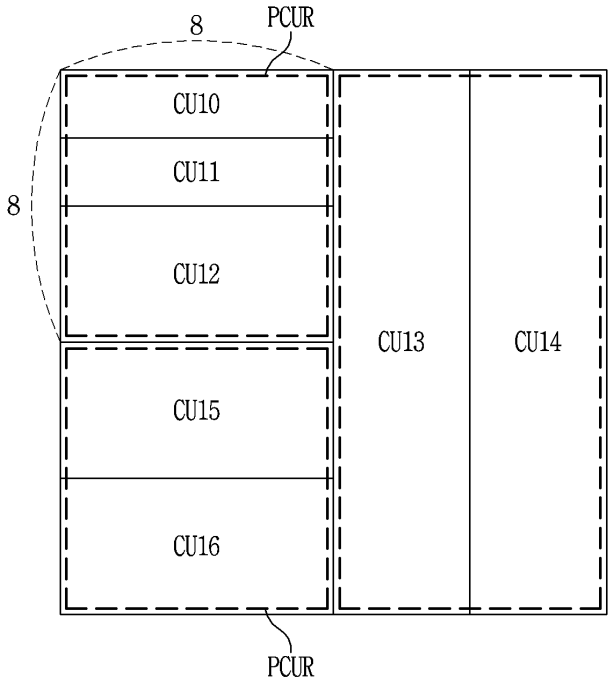
도면21


8
2101
2102
2
2
CU10
CU11
CU12
CU13
CU14
CU15
CU16
CU17

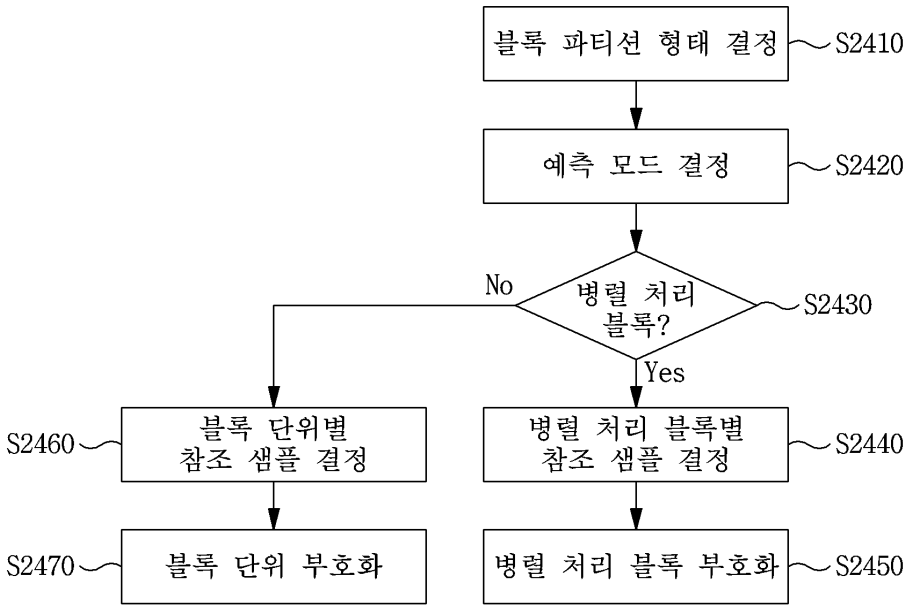
도면22


(a)
CU18
CU19
CU20
CU24
CU25
CU21
CU22
CU23
CU26
CU27
CU28
CU29
(b)
CU30
CU31
CU32
CU35
CU33
CU34
CU36
CU37
CU38
CU39
CU40
(c)
CU41
CU42
CU45
CU48
CU43
CU44
CU46
CU47
CU49
CU50
CU51
CU52
CU53
CU54
(d)
CU41
CU42
CU45
CU48
CU43
CU44
CU46
CU47
CU49
CU50
CU51
CU52
CU53
CU54

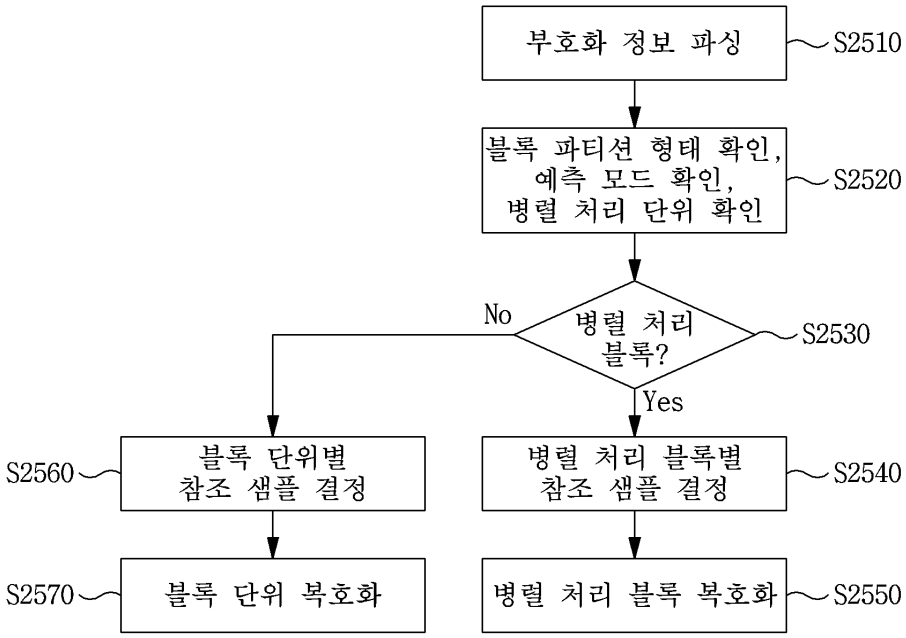
도면23


8
PCUR
CU10
CU11
CU12
CU13
CU14
CU15
CU16
8
PCUR

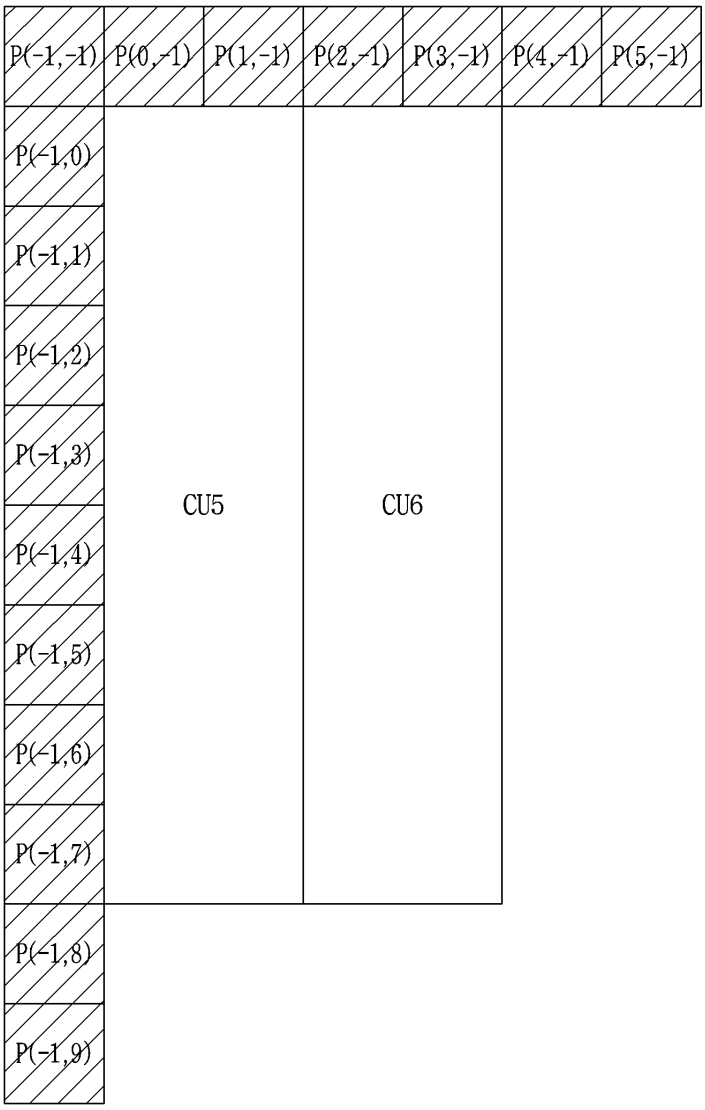
도면24


블록 파티션 형태 결정
S2410
예측 모드 결정
S2420
병렬 처리 블록?
S2430
No
Yes
블록 단위별 참조 샘플 결정
S2460
병렬 처리 블록별 참조 샘플 결정
S2440
블록 단위 부호화
S2470
병렬 처리 블록 부호화
S2450

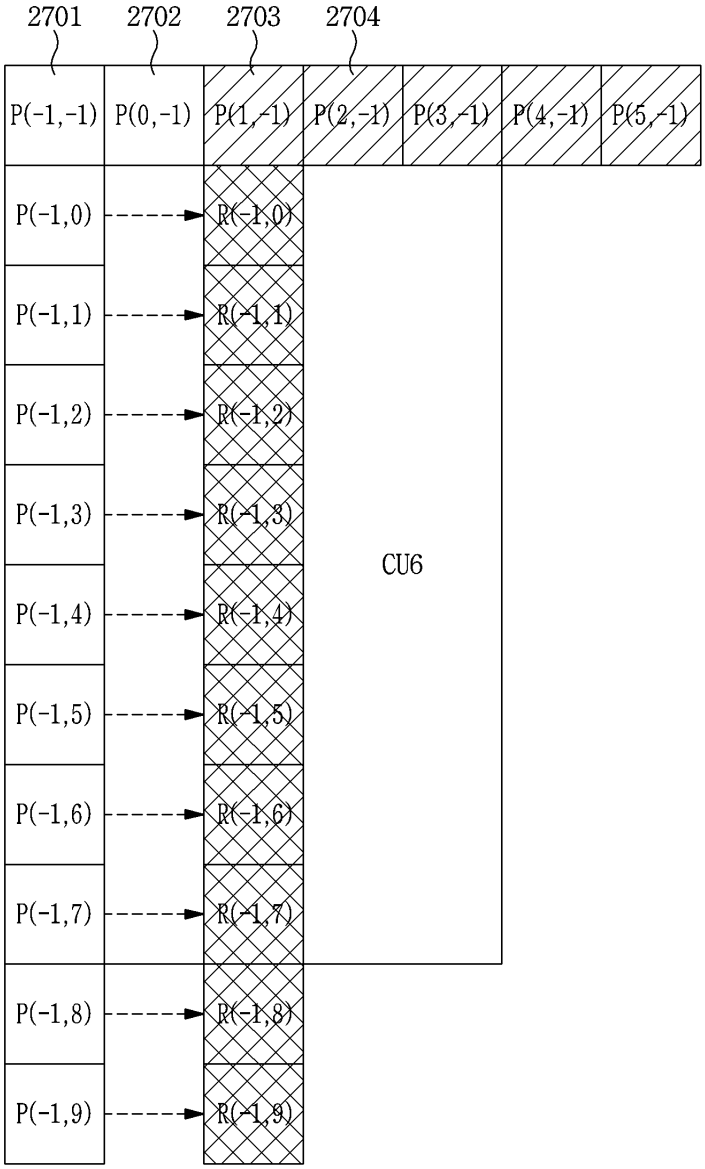
도면25


부호화 정보 파싱
S2510
블록 파티션 형태 확인,
예측 모드 확인,
병렬 처리 단위 확인
S2520
No
병렬 처리
블록?
S2530
Yes
S2560
블록 단위별
참조 샘플 결정
병렬 처리 블록별
참조 샘플 결정
S2540
S2570
블록 단위 복호화
병렬 처리 블록 복호화
S2550

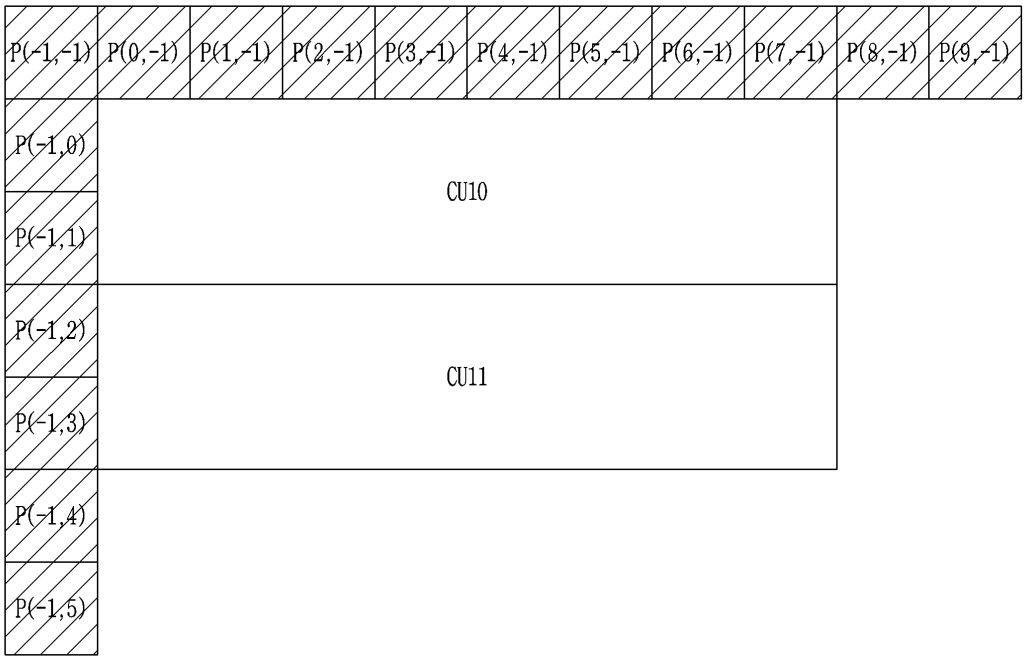
도면26


P(-1,-1)
P(0,-1)
P(1,-1)
P(2,-1)
P(3,-1)
P(4,-1)
P(5,-1)
P(-1,0)
P(-1,1)
P(-1,2)
P(-1,3)
P(-1,4)
P(-1,5)
P(-1,6)
P(-1,7)
P(-1,8)
P(-1,9)
CU5
CU6

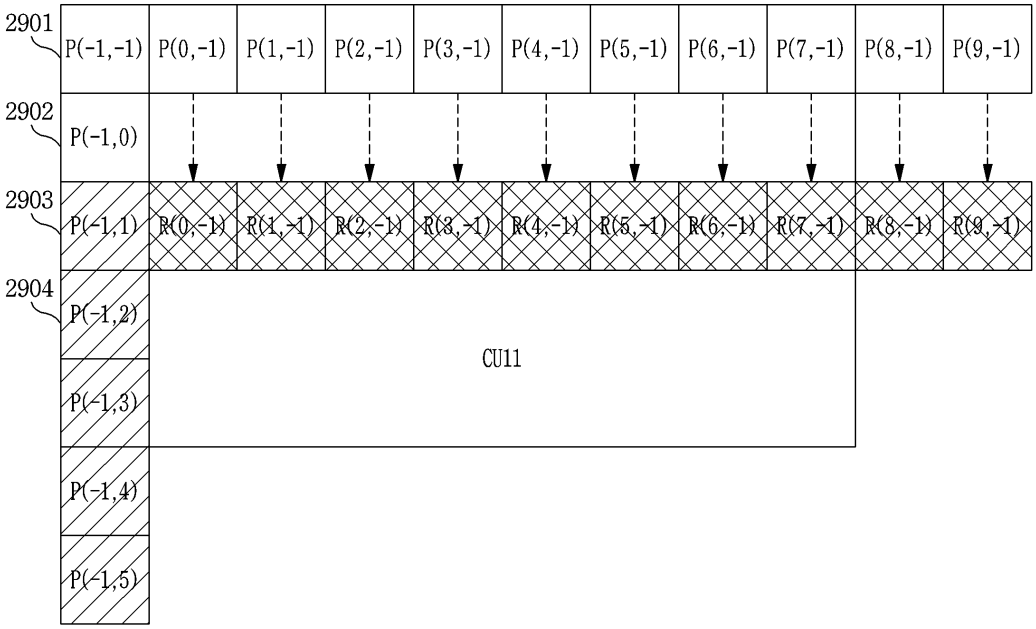
도면27


2701
2702
2703
2704
P(-1,-1)
P(0,-1)
P(1,-1)
P(2,-1)
P(3,-1)
P(4,-1)
P(5,-1)
P(-1,0)
P(-1,1)
P(-1,2)
P(-1,3)
P(-1,4)
P(-1,5)
P(-1,6)
P(-1,7)
P(-1,8)
P(-1,9)
R(-1,0)
R(-1,1)
R(-1,2)
R(-1,3)
R(-1,4)
R(-1,5)
R(-1,6)
R(-1,7)
R(-1,8)
R(-1,9)
CU6

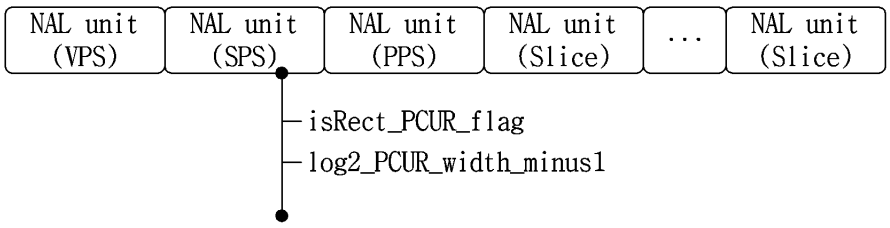
도면28


P(-1,-1)
P(0,-1)
P(1,-1)
P(2,-1)
P(3,-1)
P(4,-1)
P(5,-1)
P(6,-1)
P(7,-1)
P(8,-1)
P(9,-1)
P(-1,0)
P(-1,1)
P(-1,2)
P(-1,3)
P(-1,4)
P(-1,5)
CU10
CU11

도면29


2901
2902
2903
2904
P(-1,-1)
P(0,-1)
P(1,-1)
P(2,-1)
P(3,-1)
P(4,-1)
P(5,-1)
P(6,-1)
P(7,-1)
P(8,-1)
P(9,-1)
P(-1,0)
P(-1,1)
R(0,-1)
R(1,-1)
R(2,-1)
R(3,-1)
R(4,-1)
R(5,-1)
R(6,-1)
R(7,-1)
R(8,-1)
R(9,-1)
P(-1,2)
P(-1,3)
P(-1,4)
P(-1,5)
CU11

도면30


NAL unit (VPS)
NAL unit (SPS)
NAL unit (PPS)
NAL unit (Slice)
...
NAL unit (Slice)
isRect_PCUR_flag
log2_PCUR_width_minus1