



(19) 대한민국특허청(KR)  
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0024658  
(43) 공개일자 2022년03월03일

(51) 국제특허분류(Int. Cl.)  
H04N 19/645 (2014.01) H04N 19/129 (2014.01)  
H04N 19/13 (2014.01) H04N 19/176 (2014.01)  
H04N 19/18 (2014.01) H04N 19/91 (2014.01)  
(52) CPC특허분류  
H04N 19/645 (2015.01)  
H04N 19/129 (2015.01)  
(21) 출원번호 10-2022-7001694  
(22) 출원일자(국제) 2020년05월12일  
심사청구일자 없음  
(85) 번역문제출일자 2022년01월17일  
(86) 국제출원번호 PCT/US2020/032460  
(87) 국제공개번호 WO 2020/263442  
국제공개일자 2020년12월30일  
(30) 우선권주장  
62/865,916 2019년06월24일 미국(US)  
(뒷면에 계속)

(71) 출원인  
알리바바 그룹 홀딩 리미티드  
케이만군도, 그랜드 케이만, 조지 타운, 피.오.  
박스 847, 원 캐피탈 플레이스, 플로어 4  
(72) 발명자  
사르위, 모함메드, 지.  
미국 캘리포니아 94402 샌 마테오 스위트 400 사  
우스 엘 카미노 리얼 400 알리바바 그룹 리갈 디  
파트먼트  
예, 안  
미국 캘리포니아 94402 샌 마테오 스위트 400 사  
우스 엘 카미노 리얼 400 알리바바 그룹 리갈 디  
파트먼트  
루오, 지안콩  
미국 캘리포니아 94402 샌 마테오 스위트 400 사  
우스 엘 카미노 리얼 400 알리바바 그룹 리갈 디  
파트먼트  
(74) 대리인  
특허법인 광장리앤코

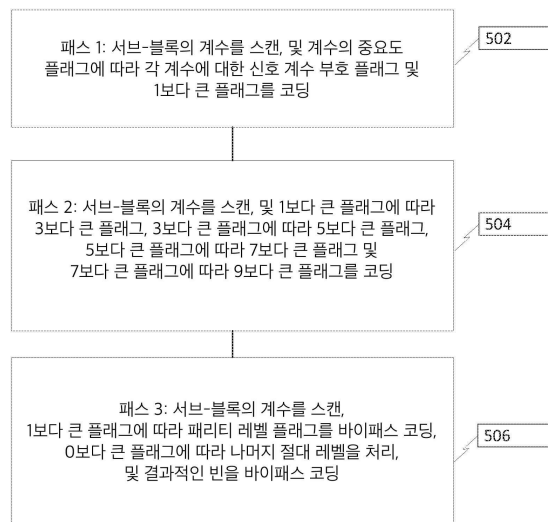
전체 청구항 수 : 총 24 항

(54) 발명의 명칭 비디오 데이터의 변환-스킵 잔차 코딩

(57) 요약

본 개시는 변환-스킵 장치 비디오 데이터 코딩 및 디코딩을 위한 시스템 및 방법을 제공한다. 일 예시적인 방법  
은: 비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하는 단계를 포함하고, 스캐닝의 제1 패  
스는: 변환 계수에 대한 패리티 레벨 플래그를 바이패스 코딩하는 단계를 포함하고, 패리티 레벨 플래그는 변환  
계수의 레벨의 절대 값의 패리티를 나타낸다.

대표도 - 도5



(52) CPC특허분류

*H04N 19/13* (2015.01)  
*H04N 19/176* (2015.01)  
*H04N 19/18* (2015.01)  
*H04N 19/91* (2015.01)

(30) 우선권주장

62/902,115 2019년09월18일 미국(US)  
62/953,460 2019년12월24일 미국(US)

## 명세서

### 청구범위

#### 청구항 1

비디오 데이터의 인코더에 의해 구현되는 코딩 방법으로서, 상기 방법은:

비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하는 단계를 포함하고, 스캐닝의 상기 제1 패스는:

상기 변환 계수에 대한 패리티 레벨 플래그를 바이패스 코딩하는 단계(bypass coding) - 상기 패리티 레벨 플래그는 상기 변환 계수의 레벨의 절대 값의 패리티를 나타냄 -를 포함하는, 코딩 방법.

#### 청구항 2

제1항에 있어서,

스캐닝의 상기 제1 패스 이전에, 상기 변환 계수의 스캐닝의 제2 패스를 수행하는 단계를 더 포함하고, 스캐닝의 상기 제2 패스는:

1보다 큰 플래그를 코딩하는 단계 - 상기 1보다 큰 플래그는 상기 절대 값이 1보다 큰지 여부를 나타냄 - 를 포함하고;

스캐닝의 상기 제1 패스는:

상기 1보다 큰 플래그가 상기 절대 값이 1보다 크다는 것을 나타내는 것에 응답하여 상기 패리티 레벨 플래그를 바이패스 코딩하는 단계를 더 포함하는, 코딩 방법.

#### 청구항 3

제2항에 있어서, 상기 서브-블록은 복수의 변환 계수를 갖고, 스캐닝의 상기 제2 패스를 수행하는 단계는:

컨텍스트 코딩된 빈(context coded bin)의 수가 최대 제한에 도달할 때까지 상기 복수의 변환 계수를 스캐닝하는 단계; 및

상기 컨텍스트 코딩된 빈의 수가 상기 최대 제한에 도달하는 것에 응답하여, 상기 제2 패스에서 스캔되지 않은 변환 계수의 레벨의 절대 값을 바이패스 코딩하는 단계 - 상기 바이패스 코딩하는 단계는 골롬-라이스 코딩(Golomb-Rice coding)을 사용하여 상기 절대 값을 이진화하는 단계를 포함함 -를 더 포함하는, 코딩 방법.

#### 청구항 4

제2항에 있어서, 스캐닝의 상기 제2 패스는:

상기 변환 계수에 대한 중요도 플래그(significance flag)를 코딩하는 단계 - 상기 중요도 플래그는 상기 변환 계수의 상기 레벨이 0인지를 나타냄 -; 및

상기 중요도 플래그가 상기 변환 계수의 상기 레벨이 0이 아님을 나타내는 것에 응답하여 상기 1보다 큰 플래그를 코딩하는 단계를 더 포함하는, 코딩 방법.

#### 청구항 5

제2항에 있어서, 스캐닝의 상기 제2 패스는:

상기 1보다 큰 플래그가 상기 절대 값이 1보다 크다는 것을 나타내는 것에 응답하여 상기 변환 계수에 대한 3보다 큰 플래그를 코딩하는 단계 - 상기 3보다 큰 플래그는 상기 절대 값이 3보다 큰지 여부를 나타냄 -를 더 포함하는, 코딩 방법.

#### 청구항 6

제5항에 있어서,

스캐닝의 상기 제2 패스 이후 및 스캐닝의 상기 제1 패스 이전에, 상기 변환 계수의 스캐닝의 제3 패스를 수행하는 단계를 더 포함하고, 스캐닝의 상기 제3 패스는:

상기 3보다 큰 플래그가 상기 절대 값이 3보다 크다는 것을 나타내는 것에 응답하여, 상기 변환 계수에 대한 5보다 큰 플래그를 코딩하는 단계 - 상기 5보다 큰 플래그는 상기 절대 값이 5보다 큰지 여부를 나타냄 -;

상기 5보다 큰 플래그가 상기 절대 값이 5보다 크다는 것을 나타내는 것에 응답하여, 상기 변환 계수에 대한 7보다 큰 플래그를 코딩하는 단계 - 상기 7보다 큰 플래그는 상기 절대 값이 7보다 큰지 여부를 나타냄 -; 및

상기 7보다 큰 플래그가 상기 절대 값이 7보다 크다는 것을 나타내는 것에 응답하여, 상기 변환 계수에 대한 9보다 큰 플래그를 코딩하는 단계 - 상기 9보다 큰 플래그는 상기 절대 값이 9보다 큰지 여부를 나타냄 -를 더 포함하는, 코딩 방법.

#### 청구항 7

제6항에 있어서, 스캐닝의 상기 제1 패스는:

상기 9보다 큰 플래그가 상기 절대 값이 9보다 크다는 것을 나타내는 것에 응답하여 상기 변환 계수에 대한 나머지 절대 레벨 플래그를 코딩하는 단계를 더 포함하고, 상기 나머지 절대 레벨 플래그는 상기 변환 계수의 상기 레벨의 나머지 절대 값을 나타내는, 코딩 방법.

#### 청구항 8

제2항에 있어서,

스캐닝의 상기 제2 패스 이후 및 스캐닝의 상기 제1 패스 이전에, 상기 변환 계수의 스캐닝의 제3 패스를 수행하는 단계를 더 포함하고, 스캐닝의 상기 제3 패스는:

상기 1보다 큰 플래그가 상기 절대 값이 1보다 크다는 것을 나타내는 것에 응답하여 상기 변환 계수에 대한 3보다 큰 플래그를 코딩하는 단계 - 상기 3보다 큰 플래그는 상기 절대 값이 3보다 큰지 여부를 나타냄 -를 더 포함하는, 코딩 방법.

#### 청구항 9

제8항에 있어서, 스캐닝의 상기 제3 패스는:

상기 3보다 큰 플래그가 상기 절대 값이 3보다 크다는 것을 나타내는 것에 응답하여, 상기 변환 계수에 대한 5보다 큰 플래그를 코딩하는 단계 - 상기 5보다 큰 플래그는 상기 절대 값이 5보다 큰지 여부를 나타냄 -;

상기 5보다 큰 플래그가 상기 절대 값이 5보다 크다는 것을 나타내는 것에 응답하여, 상기 변환 계수에 대한 7보다 큰 플래그를 코딩하는 단계 - 상기 7보다 큰 플래그는 상기 절대 값이 7보다 큰지 여부를 나타냄 -; 및

상기 7보다 큰 플래그가 상기 절대 값이 7보다 크다는 것을 나타내는 것에 응답하여, 상기 변환 계수에 대한 9보다 큰 플래그를 코딩하는 단계 - 상기 9보다 큰 플래그는 상기 절대 값이 9보다 큰지 여부를 나타냄 -를 더 포함하는, 코딩 방법.

#### 청구항 10

제9항에 있어서, 스캐닝의 상기 제1 패스는:

상기 9보다 큰 플래그가 상기 절대 값이 9보다 크다는 것을 나타내는 것에 응답하여 상기 변환 계수에 대한 나머지 절대 레벨 플래그를 코딩하는 단계를 더 포함하고, 상기 나머지 절대 레벨 플래그는 상기 변환 계수의 상기 레벨의 나머지 절대 값을 나타내는, 코딩 방법.

#### 청구항 11

제1항에 있어서, 상기 서브-블록은 복수의 변환 계수를 갖고, 스캐닝의 상기 제1 패스는 상기 복수의 변환 계수를 역순으로 스캔함으로써 수행되는, 코딩 방법.

## 청구항 12

제11항에 있어서,

스캐닝의 상기 제1 패스를 수행하기 이전에 상기 복수의 변환 계수를 플립하는 단계(flippping)를 더 포함하는, 코딩 방법.

## 청구항 13

제1항에 있어서, 상기 코딩 방법은 변환-스킵 잔차 코딩 방법(transform-skip residual coding method)인, 코딩 방법.

## 청구항 14

제1항에 있어서,

상기 비디오 프레임을 수신하는 단계; 및

상기 비디오 프레임을 복수의 서브-블록으로 분할하는 단계를 더 포함하는, 코딩 방법.

## 청구항 15

비디오 데이터의 디코더에 의해 구현되는 디코딩 방법에 있어서, 상기 방법은:

비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하는 단계를 포함하고, 스캐닝의 상기 제1 패스는:

상기 변환 계수에 대한 패리티 레벨 플래그를 바이패스 디코딩하는 단계 - 상기 패리티 레벨 플래그는 상기 변환 계수의 레벨의 절대 값의 패리티를 나타냄 -를 포함하는, 디코딩 방법.

## 청구항 16

제15항에 있어서,

스캐닝의 상기 제1 패스 이전에, 상기 변환 계수의 스캐닝의 제2 패스를 수행하는 단계를 더 포함하고 - 스캐닝의 상기 제2 패스는:

1보다 큰 플래그를 디코딩하는 단계 - 상기 1보다 큰 플래그는 상기 절대 값이 1보다 큰지 여부를 나타냄 -를 포함하고;

스캐닝의 상기 제1 패스는:

상기 1보다 큰 플래그가 상기 절대 값이 1보다 크다는 것을 나타내는 것에 응답하여, 상기 제1 패스에서 상기 패리티 레벨 플래그를 바이패스 디코딩하는 단계를 더 포함하는, 디코딩 방법.

## 청구항 17

제16항에 있어서, 스캐닝의 상기 제2 패스는:

상기 변환 계수에 대한 중요도 플래그를 디코딩하는 단계 - 상기 중요도 플래그는 상기 변환 계수의 상기 레벨이 0인지를 나타냄 -; 및

상기 중요도 플래그가 상기 변환 계수의 상기 레벨이 0이 아님을 나타내는 것에 응답하여 상기 1보다 큰 플래그를 디코딩하는 단계를 더 포함하는, 디코딩 방법.

## 청구항 18

제16항에 있어서, 스캐닝의 상기 제1 패스는:

상기 1보다 큰 플래그가 상기 절대 값이 1보다 크다는 것을 나타내는 것에 응답하여 상기 변환 계수에 대한 3보다 큰 플래그를 디코딩하는 단계 - 상기 3보다 큰 플래그는 상기 절대 값이 3보다 큰지 여부를 나타냄 -;

상기 3보다 큰 플래그가 상기 절대 값이 3보다 크다는 것을 나타내는 것에 응답하여 상기 변환 계수에 대한 5보

다 큰 플래그를 디코딩하는 단계 - 상기 5보다 큰 플래그는 상기 절대 값이 5보다 큰지 여부를 나타냄 -;

상기 5보다 큰 플래그가 상기 절대 값이 5보다 크다는 것을 나타내는 것에 응답하여 상기 변환 계수에 대한 7보다 큰 플래그를 디코딩하는 단계 - 상기 7보다 큰 플래그는 상기 절대 값이 7보다 큰지 여부를 나타냄 -; 및

상기 7보다 큰 플래그가 상기 절대 값이 7보다 크다는 것을 나타내는 것에 응답하여 상기 변환 계수에 대한 9보다 큰 플래그를 디코딩하는 단계 - 상기 9보다 큰 플래그는 상기 절대 값이 9보다 큰지 여부를 나타냄 -를 더 포함하는, 디코딩 방법.

#### 청구항 19

제18항에 있어서,

상기 변환 계수의 스캐닝의 제3 패스를 수행하는 단계를 더 포함하고, 스캐닝의 상기 제3 패스는:

상기 9보다 큰 플래그가 상기 절대 값이 9보다 크다는 것을 나타내는 것에 응답하여 상기 변환 계수에 대한 나머지 절대 레벨 플래그를 디코딩하는 단계를 포함하고, 상기 나머지 절대 레벨 플래그는 상기 변환 계수의 상기 레벨의 나머지 절대 값을 나타내는, 디코딩 방법.

#### 청구항 20

제15항에 있어서, 상기 서브-블록은 복수의 변환 계수를 갖고, 스캐닝의 상기 제1 패스는 상기 서브-블록의 우측-하단 코너로부터 상기 서브-블록의 좌측-상단 코너까지 상기 복수의 변환 계수를 스캔함으로써 수행되는, 디코딩 방법.

#### 청구항 21

제20항에 있어서,

스캐닝의 상기 제1 패스를 수행하기 이전에 상기 복수의 변환 계수를 플립하는 단계를 더 포함하는, 디코딩 방법.

#### 청구항 22

제15항에 있어서, 상기 디코딩 방법은 변환-스킵 잔차 디코딩 방법인, 디코딩 방법.

#### 청구항 23

비디오 데이터를 인코딩하기 위한 시스템으로서, 상기 시스템은,

명령어의 세트를 저장하는 메모리; 및

프로세서를 포함하고, 상기 프로세서는 상기 명령어의 세트를 실행하여, 상기 시스템이:

비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하게 하도록 구성되고, 스캐닝의 상기 제1 패스는:

상기 변환 계수에 대한 패리티 레벨 플래그를 바이패스 코딩하는 것 - 상기 패리티 레벨 플래그는 상기 변환 계수의 레벨의 절대 값의 패리티를 나타냄 -을 포함하는, 시스템.

#### 청구항 24

비디오 데이터를 디코딩하기 위한 시스템으로서,

명령어의 세트를 저장하는 메모리; 및

프로세서를 포함하고, 상기 프로세서는 상기 명령어의 세트를 실행하여, 상기 시스템이:

비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하게 하도록 구성되고, 스캐닝의 상기 제1 패스는:

상기 변환 계수에 대한 패리티 레벨 플래그를 디코딩하는 것 - 상기 패리티 레벨 플래그는 상기 변환 계수의 레벨의 절대 값의 패리티를 나타냄 -을 포함하는, 시스템.

## 발명의 설명

### 기술 분야

[0001] 연관된 출원에 대한 상호 참조

[0002] 본 개시는 2019년 6월 24일에 출원된 미국 가특허출원 제62/865,916호; 2019년 9월 18일에 출원된 미국 가특허출원 제62/902,115호; 및 2019년 12월 24일에 출원된 미국 가특허출원 제62/953,460호에 대한 우선권 및 우선권의 이익을 주장한다. 세 개의 가출원은 그 전체가 본원에 참조로 통합된다.

[0003] 기술분야

[0004] 본 개시는 일반적으로 비디오 데이터 처리에 관한 것이고, 보다 구체적으로 비디오 데이터의 변환-스킵 잔차 코딩(transform-skip residual coding)에 관한 것이다.

### 배경 기술

[0005] 비디오 압축 및 압축 해제 산업에서 비디오 코딩에 대한 새로운 표준이 개발되고 있다. 예를 들어, ITU-T 비디오 코딩 전문가 그룹(Video Coding Expert Group, "VCEG") 및 ISO/IEC 동영상 전문가 그룹(Moving Picture Expert Group, "MPEG")의 공동 비디오 전문가 팀(Joint Video Experts Team, "JVET")은 현재 다용도 비디오 코딩(Versatile Video Coding, "VVC") 표준을 개발하고 있다. VVC 표준은 그의 이전 버전 - 고효율 비디오 코딩("HEVC/H.265") 표준의 압축 효율성을 두 배로 높이는 것을 목표로 한다. 즉, VVC의 목표는 절반의 대역폭을 사용하여 HEVC/H.265와 동일한 주관적인 품질을 달성하는 것이다.

### 발명의 내용

[0006] 본 개시의 실시예는 변환-스킵 잔차 비디오 데이터 코딩을 위한 방법 및 시스템을 제공한다.

[0007] 일 예시적인 방법은: 비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하는 단계를 포함하고, 스캐닝의 제1 패스는 변환 계수에 대한 패리티 레벨 플래그를 바이패스 코딩하는 것(bypass coding)을 포함하고, 패리티 레벨 플래그는 변환 계수의 레벨의 절대 값의 패리티를 나타낸다.

[0008] 다른 예시적인 방법은: 비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하는 단계를 포함하고, 스캐닝의 제1 패스는 변환 계수에 대한 패리티 레벨 플래그를 바이패스 디코딩하는 것을 포함하고, 패리티 레벨 플래그는 변환 계수의 레벨의 절대 값의 패리티를 나타낸다.

[0009] 일 예시적인 시스템은: 명령어의 세트를 저장하는 메모리; 및 프로세서를 포함하고, 프로세서는 명령어의 세트를 실행하여 시스템이: 비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하게 하도록 구성되고, 스캐닝의 제1 패스는 변환 계수에 대한 패리티 레벨 플래그를 바이패스 코딩하는 것을 포함하고, 패리티 레벨 플래그는 변환 계수의 레벨의 절대 값의 패리티를 나타낸다.

[0010] 다른 예시적인 시스템은: 명령어의 세트를 저장하는 메모리; 및 프로세서를 포함하고, 프로세서는 명령어의 세트를 실행하여 시스템이: 비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하게 하도록 구성되고, 스캐닝의 제1 패스는 변환 계수에 대한 패리티 레벨 플래그를 바이패스 디코딩하는 것을 포함하고, 패리티 레벨 플래그는 변환 계수의 레벨의 절대 값의 패리티를 나타낸다.

### 도면의 간단한 설명

[0011] 본 개시의 실시예 및 다양한 양상은 다음의 상세한 설명 및 첨부 도면에 예시된다. 도면에 도시된 다양한 피쳐(feature)는 축척대로 도시된 것은 아니다.

도 1은 하이브리드 비디오 코딩 시스템(hybrid video coding system)의 예시적인 인코더 블록도를 예시한다.

도 2는 하이브리드 비디오 코딩 시스템의 예시적인 디코더 블록도를 예시한다.

도 3(도 3a-3d)은 코딩을 변환하기 위한 신택스(syntax)를 포함하는 예시적인 의사코드(pseudocode)를 예시한다.

도 4(도 4a-4b)는 스킵 잔차 코딩을 변환하는 신택스를 포함하는 예시적인 의사코드를 예시한다.

도 5는 본 개시의 일부 실시예에 따른 코딩 패스의 수가 3으로 감소되는 변환-스킵 잔차 코딩의 예시적인 방법을 예시한다.

도 6(도 6a-6b)은 본 개시의 일부 실시예에 따른 도 5에 도시된 방법의 선택스를 포함하는 예시적인 의사코드를 예시한다.

도 7은 본 개시의 일부 실시예에 따른, 코딩 패스의 수가 3으로 감소되는 변환-스킵 잔차 코딩의 다른 예시적인 방법을 예시한다.

도 8(도 8a-8b)은 본 개시의 일부 실시예에 따른, 도 7에 도시된 방법의 선택스를 포함하는 예시적인 의사코드를 예시한다.

도 9는 본 개시의 일부 실시예에 따른, 8x8 변환-스킵 블록의 예시적인 역방향 스캐닝을 예시한다.

도 10a는 본 개시의 일부 실시예에 따른, 플립되기(flipped) 이전의 8x8 블록을 예시한다.

도 10b는 본 개시의 일부 실시예에 따른, 도 10a에서의 8x8 블록이 플립된 이후의 결과적인 블록을 예시한다.

도 11은 본 개시의 일부 실시예에 따른, 예시적인 다중-패스 코딩을 예시한다.

도 12는 본 개시의 일부 실시예에 따른, 레벨의 절대 값에 대한 예시적인 단일 패스 바이패스 코딩 방법(bypass coding method)을 예시한다.

도 13은 본 개시의 일부 실시예에 따른, 라이스 파라미터(Rice parameters)에 대한 예시적인 룩업 테이블(look-up table)을 예시한다.

도 14는 본 개시의 일부 실시예에 따른, 단일 패스 바이패스 코딩과 결합된, 코딩 패스의 수가 3으로 감소된 변환-스킵 잔차 코딩의 예시적인 방법을 예시한다.

도 15(도 15a-15b)는 본 개시의 일부 실시예에 따른, 도 14에서의 방법과 결합된 바이패스 코딩의 선택스를 포함하는 예시적인 의사코드를 예시한다.

도 16은 본 개시의 일부 실시예에 따른, 컨텍스트 코딩에 대한 제1 패스 및 골롬-라이스 코딩(Golomb-Rice coding)에 대한 제2 패스를 갖는 변환-스킵 잔차 코딩의 예시적인 방법을 예시한다.

도 17(도 17a-17c)은 본 개시의 일부 실시예에 따른 도 16에서의 방법의 바이패스 코딩의 선택스를 포함하는 예시적인 의사코드를 예시한다.

도 18은 본 개시의 일부 실시예에 따른, 최소 바이패스 코딩된 값이 0과 동일할 때 라이스 파라미터에 대한 예시적인 룩업 테이블을 예시한다.

### 발명을 실시하기 위한 구체적인 내용

[0012] 이제 예시적인 실시예에 대한 참조가 상세히 이루어질 것이며, 이의 예시는 첨부 도면에 예시된다. 다음 설명은 달리 나타나지 않는 한 상이한 도면에서 동일한 번호가 동일하거나 유사한 요소를 나타내는 첨부 도면을 참조한다. 예시적인 실시예의 다음 설명에서 제시된 구현은 본 발명과 일치하는 모든 구현을 나타내는 것은 아니다. 대신에, 이들은 첨부된 청구범위에서 인용된 본 발명에 관련된 양상과 일치하는 장치 및 방법의 예시일 뿐이다. 본 개시의 특정 양상은 아래에서 더 상세히 설명된다. 본원에 제공된 용어 및 정의는 참조로 통합된 용어 및/또는 정의와 상충하는 경우 제어된다.

[0013] 비디오는 시각적 정보를 저장하기 위해 시간적인 시퀀스로 배열된 정적 픽처(또는 "프레임")의 세트이다. 비디오 캡처 디바이스(예를 들어, 카메라)는 이 픽처를 시간적인 시퀀스로 캡처하고 저장하는 데 사용될 수 있고, 비디오 재생 디바이스(예를 들어, 디스플레이의 기능을 갖는 텔레비전, 컴퓨터, 스마트폰, 태블릿 컴퓨터, 비디오 플레이어 또는 임의의 최종-사용자 단말기)는 이러한 픽처를 시간적인 시퀀스로 디스플레이하기 위해 사용될 수 있다. 또한, 일부 응용에서, 비디오 캡처링 디바이스는 감시, 회의 또는 실시간 방송과 같이 캡처된 비디오를 실시간으로 비디오 재생 디바이스(예를 들어, 모니터를 갖는 컴퓨터)에 송신할 수 있다.

[0014] 이러한 응용에 의해 필요한 저장 공간 및 송신 대역폭을 감소시키기 위해, 비디오는 압축될 수 있다. 예를 들어, 비디오는 저장 및 송신 이전에 압축되고, 디스플레이 이전에 압축 해제될 수 있다. 압축 및 압축 해제는 프로세서(예를 들어, 일반 컴퓨터의 프로세서) 또는 특수 하드웨어에 의해 실행되는 소프트웨어에 의해 구현될 수 있다. 압축을 위한 모듈은 일반적으로 "인코더"로 지칭되고, 압축 해제를 위한 모듈은 일반적으로 "디코더"

로 지칭된다. 인코더 및 디코더를 집합적으로 "코덱(codec)"으로 지칭될 수 있다. 인코더 및 디코더는 임의의 다양한 적합한 하드웨어, 소프트웨어, 또는 이들의 조합 중 임의의 것으로 구현될 수 있다. 예를 들어, 인코더 및 디코더의 하드웨어 구현은 하나 이상의 마이크로프로세서, 디지털 신호 프로세서(DSP), 애플리케이션-특정 집적 회로(application-specific integrated circuits, ASIC), 필드 프로그램 가능 게이트 어레이(Field-Programmable Gate Array, FPGA), 이산 로직 또는 임의의 임의의 조합과 같은 회로를 포함할 수 있다. 인코더 및 디코더의 소프트웨어 구현은 컴퓨터 판독 가능 매체에 고정된 프로그램 코드, 컴퓨터-실행가능 명령어, 펌웨어 또는 임의의 적합한 컴퓨터 구현 알고리즘 또는 프로세스를 포함할 수 있다. 비디오 압축 및 압축 해제에는 MPEG-1, MPEG-2, MPEG-4, H.26x 시리즈 등과 같은 다양한 알고리즘 또는 표준에 의해 구현될 수 있다. 일부 응용에서, 코덱은 제1 코딩 표준으로부터 비디오를 압축 해제하고 제2 코딩 표준을 사용하여 압축 해제된 비디오를 재압축할 수 있으며, 이 경우 코덱은 "트랜스코더(transcoder)"로 지칭될 수 있다.

[0015] 비디오 인코딩 프로세스는 픽처를 재구성하는 데 사용될 수 있는 유용한 정보를 식별하고 유지할 수 있다. 비디오 인코딩 프로세스에서 무시된 정보가 완전히 재구성될 수 없는 경우, 인코딩 프로세스는 "손실이 있는 것(lossy)"으로 지칭될 수 있다. 그렇지 않으면, "무손실"로 지칭될 수 있다. 대부분의 인코딩 프로세스는 손실이 있고, 이는 필요한 저장 공간 및 송신 대역폭을 감소시키기 위한 절충안이다.

[0016] 다수의 경우에서, 인코딩되는 픽처("현재 픽처"로 지칭됨)의 유용한 정보는 참조 픽처(예를 들어, 이전에 인코딩되거나 재구성된 픽처)에 대한 변경을 포함할 수 있다. 이러한 변경은 픽셀의 위치 변경, 광도 변경 또는 색상 변경을 포함할 수 있으며, 그 중 위치 변경이 주로 관련된다. 객체를 나타내는 픽셀의 그룹의 위치 변경은 참조 픽처와 현재 픽처 사이의 객체의 움직임을 반영할 수 있다.

[0017] 절반의 대역폭을 사용하여 HEVC/H.265와 동일한 주관적인 품질을 달성하기 위해, JVET는 공동 탐색 모델("JEM(joint exploration model)") 참조 소프트웨어를 사용하여 HEVC를 넘는 기술을 개발하고 있다. 코딩 기술이 JEM에 통합됨에 따라, JEM은 HEVC보다 상당히 더 높은 코딩 성능을 달성하였다. VCEG 및 MPEG는 또한, HEVC를 넘는 차세대 비디오 압축 표준의 개발을 공식적으로 시작하였다.

[0018] VVC 표준은 더 나은 압축 성능을 제공하는 더 많은 코딩 기술을 계속 포함하고 있다. VVC는 HEVC, H.264/AVC, MPEG2, H.263 등과 같은 최신 비디오 압축 표준에서 사용된 것과 동일한 하이브리드 비디오 코딩 시스템을 기초로 한다. 도 1은 하이브리드 비디오 코딩 시스템(hybrid video coding system)의 예시적인 인코더 블록도를 예시한다. 도 1에 도시된 바와 같이, 비디오 인코더(200)는 비디오 블록, 또는 비디오 블록의 파티션 또는 서브-파티션을 포함하는, 비디오 프레임 내의 블록의 인트라-코딩(intra-coding) 또는 인터-코딩(inter-coding)을 수행할 수 있다. 인트라-코딩은 주어진 비디오 프레임 내에서 비디오에서의 공간 중복성을 감소시키거나 또는 제거하기 위해 공간 예측에 의존할 수 있다. 인터-코딩은 비디오 시퀀스의 인접 프레임 내에서 비디오에서의 시간적 중복성을 감소 또는 제거하기 위해 시간 예측에 의존할 수 있다. 인트라 모드는 다수의 공간 기반 압축 모드를 지칭할 수 있다. (단방향-예측 또는 양방향 예측과 같은) 인터 모드는 다수의 시간-기반 압축 모드를 지칭할 수 있다.

[0019] 도 1을 참조하면, 입력 비디오 신호(202)는 한 블록씩 처리될 수 있다. 예를 들어, 비디오 블록 유닛은  $16 \times 16$  픽셀 블록(예를 들어, 매크로블록(MB))일 수 있다. 비디오 블록 유닛의 크기는 사용된 코딩 기술, 요구되는 정확도 및 효율성에 의존하여 달라질 수 있다. HEVC에서, 확장된 블록 크기(예를 들어, 코딩 트리 유닛(coding tree unit, CTU))은 해상도, 예를 들어 1080p 및 그 이상의 비디오 신호를 압축하기 위해 사용될 수 있다. HEVC에서, CTU는 크로마 샘플에 대응하는 최대  $64 \times 64$  루마 샘플(luma samples), 및 연관된 선택 요소들을 포함할 수 있다. VVC에서, CTU의 크기는  $128 \times 128$  루마 샘플, 대응하는 크로마 샘플(chroma samples), 및 연관된 선택 요소들을 포함하도록 더 증가될 수 있다. CTU는 예를 들어, 쿼드-트리(quad-tree), 이진 트리 또는 삼항 트리(ternary tree)를 사용하여 코딩 유닛(CU)으로 더 나눌 수 있다. CU는 개별적인 예측 방법이 적용될 수 있는 예측 유닛(PU)으로 더 분할될 수 있다. 각 입력 비디오 블록은 공간 예측 유닛(260) 또는 시간 예측 유닛(262)을 사용하여 처리될 수 있다.

[0020] 공간 예측 유닛(260)은 현재 블록을 포함하는 동일한 픽처/슬라이스에 대한 정보를 사용하여 현재 블록/CU에 대한 공간 예측(예를 들어, 인트라 예측)을 수행한다. 공간 예측은 현재 비디오 블록을 예측하기 위해 동일한 비디오 픽처 프레임/슬라이스에서 이미 코딩된 이웃 블록으로부터의 픽셀을 사용할 수 있다. 공간 예측은 비디오 신호에 내재된 공간 중복성을 감소시킬 수 있다.

[0021] 시간 예측 유닛(262)은 현재 블록을 포함하는 픽처/슬라이스와 상이한 픽처(들)/슬라이스(들)로부터의 정보를 사용하여 현재 블록에 대한 시간 예측(예를 들어, 인터 예측)을 수행한다. 비디오 블록에 대한 시간 예측은 하

나 이상의 움직임 벡터에 의해 시그널링될 수 있다. 유닛-방향 시간 예측에서, 현재 블록에 대한 예측 신호를 생성하기 위해 하나의 참조 픽처를 나타내는 하나의 움직임 벡터만이 사용된다. 다른 한편으로, 양방향 시간 예측에서, 각각 각각의 참조 픽처를 나타내는 두 개의 움직임 벡터는 현재 블록에 대한 예측 신호를 생성하기 위해 사용될 수 있다. 움직임 벡터는 참조 프레임에서 현재 블록과 하나 이상의 연관된 블록(들) 사이의 움직임의 양 및 방향을 나타낼 수 있다. 다수의 참조 픽처가 지원되는 경우, 비디오 블록에 대해 하나 이상의 참조 픽처 인덱스가 전송될 수 있다. 하나 이상의 참조 인덱스는 참조 픽처 저장소 또는 디코딩된 픽처 버퍼(decoded picture buffer, DPB)(264)에서의 어느 참조 픽처(들)로부터 시간 예측 신호가 올 수 있는지를 식별하기 위해 사용될 수 있다.

[0022] 인코더에서의 모드 결정 및 인코더 제어 유닛(280)은 예를 들어, 레이트-왜곡 최적화(rate-distortion optimization)에 기초하여 예측 모드를 선택할 수 있다. 결정된 예측 모드에 기초하여, 예측 블록이 획득될 수 있다. 예측 블록은 가산기(216)에서 현재 비디오 블록으로부터 감산될 수 있다. 예측 잔차는 변환 유닛(204)에 의해 변환되고 양자화 유닛(206)에 의해 양자화될 수 있다. 양자화된 잔차 계수는 재구성된 잔차를 형성하기 위해, 역양자화 유닛(210)에서 역양자화되고 역변환 유닛(212)에서 역변환된다. 재구성된 잔차는 재구성된 비디오 블록을 형성하기 위해 가산기(226)에서 예측 블록에 가산될 수 있다. 루프-필터링 이전의 재구성된 비디오 블록은 인트라 예측을 위한 참조 샘플을 제공하기 위해 사용될 수 있다.

[0023] 재구성된 비디오 블록은 루프 필터(266)에서 루프 필터링을 거칠 수 있다. 예를 들어, 디블록킹 필터(deblocking filter), 샘플 적응형 오프셋(Sample Adaptive Offset, SAO), 적응형 루프 필터(Adaptive Loop Filter, ALF)와 같은 루프 필터링이 적용될 수 있다. 루프 필터링 이후의 재구성된 블록은 참조 픽처 저장소(264)에 저장될 수 있고, 다른 비디오 블록을 코딩하기 위한 인터 예측 참조 샘플을 제공하기 위해 사용될 수 있다. 출력 비디오 비트스트림(220)을 형성하기 위해, 코딩 모드(예를 들어, 인터 또는 인트라), 예측 모드 정보, 움직임 정보 및 양자화된 잔차 계수는 비트스트림(220)을 형성하기 위해 데이터가 압축되고 패킹되기 전에, 비트 레이트를 더 감소시키기 위해 엔트로피 코딩 유닛(208)으로 전송될 수 있다.

[0024] 도 2는 하이브리드 비디오 코딩 시스템의 예시적인 디코더 블록도를 예시한다. 도 2에 도시된 바와 같이, 비디오 비트스트림(302)은 엔트로피 디코딩 유닛(308)에서 언팩되거나(unpacked) 엔트로피 디코딩될 수 있다. 코딩 모드 정보는 공간 예측 유닛(360) 또는 시간 예측 유닛(362)이 선택되어야 하는지를 결정하는 데 사용될 수 있다. 예측 모드 정보는 예측 블록을 생성하기 위해 대응하는 예측 유닛으로 전송될 수 있다. 예를 들어, 움직임 보상된 예측은 시간 예측 블록을 형성하기 위해 시간 예측 유닛(362)에 의해 적용될 수 있다.

[0025] 잔차 계수는 재구성된 잔차를 획득하기 위해 역양자화 유닛(310) 및 역변환 유닛(312)으로 전송될 수 있다. 예측 블록 및 재구성된 잔차는 루프 필터링 이전에 재구성된 블록을 형성하기 위해 326에서 함께 가산될 수 있다. 그 후, 재구성된 블록은 루프 필터(366)에서 루프 필터링을 거칠 수 있다. 예를 들어, 디블록킹 필터, SAO 및 ALF와 같은 루프 필터링이 적용될 수 있다. 루프 필터링 이후에 재구성된 블록은 참조 픽처 저장소(364)에 저장될 수 있다. 참조 픽처 저장소(364)에서의 재구성된 데이터는 디코딩된 비디오(320)를 획득하기 위해 사용되거나, 또는 추후의 비디오 블록을 예측하기 위해 사용될 수 있다. 디코딩된 비디오(320)는 시스템(100)(도 1)에서 설명된 바와 같이 디스플레이 디바이스(146)와 같은 디스플레이 디바이스 상에 디스플레이될 수 있다.

[0026] VVC(예를 들어, VVC 5)에서, 블록은 변환 계수의  $M \times N$  어레이일 수 있다. 변환 계수는 변환에서 특정 1차원 또는 2차원 주파수 인덱스와 연관된 주파수 영역에 있는 것으로 간주되는 스칼라 양일 수 있다. 변환 계수 레벨은 어레이 TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ]로 나타낼 수 있다. 어레이 인덱스 x0, y0은 픽처의 좌측-상단 루마 샘플에 대해 고려된 변환 블록의 좌측-상단 루마 샘플의 위치( x0, y0 )를 지정할 수 있다. 어레이 인덱스 cIdx는 색상 구성요소에 대한 표시자를 지정할 수 있다. 어레이 인덱스 xC 및 yC는 현재 변환 블록 내에서 변환 계수 위치( xC, yC )를 지정할 수 있다.

[0027] VVC(예를 들어, VVC 5)에서, 코딩 블록의 변환 계수는 비-중첩 계수 그룹(또는 서브-블록)을 사용하여 코딩된다. 각 서브-블록에 대해, 정규(또는 컨텍스트) 코딩된 빈(bins) 및 바이패스 코딩된 빈이 코딩 순서로 분리된다. 예를 들어, 먼저, 서브-블록에 대한 모든 정규 코딩된 빈이 송신되고, 그 후에 바이패스 코딩된 빈이 송신된다. 서브-블록의 변환 계수 레벨은 스캔 위치에 대해 세 개의 패스로 코딩된다. 변환 계수 레벨은 변환 계수의 값일 수 있다. 컨텍스트 코딩에 대해, 각 빈은 컨텍스트에 의해 선택된 확률 모델을 가질 수 있다. 컨텍스트는 이전에 인코딩된 선택 요소로 지칭할 수 있다. 바이패스 코딩에 대해, 코딩 효율성의 무시할만한 손실로, 코딩 프로세스의 속도를 높이기 위해 특정 빈이 선택될 수 있다. 바이패스 코딩에서, 빈은 설정된 확률(예를 들어, 0.5와 동일한 확률)로 인코딩될 수 있다.

- [0028] 패스 1에서, 중요도 플래그(예를 들어, sig\_coeff\_flag), 1보다 큰 플래그(예를 들어, gt1\_flag), 패리티 플래그(예를 들어, par\_level\_flag) 및 3보다 큰 플래그(예를 들어, gt3\_flag)가 순서대로 코딩된다. 중요도 플래그가 1과 동일한 경우, 먼저 1보다 큰 플래그가 코딩된다. 1보다 큰 플래그는 절대 레벨(예를 들어, 레벨의 절대 값)이 1보다 큰지 여부를 지정한다. 1보다 큰 플래그가 1과 동일한 경우, 패리티 플래그 및 3보다 큰 플래그가 코딩된다. 패리티 플래그는 절대 레벨에서 2를 뺀 패리티를 지정한다. 3보다 큰 플래그는 절대 레벨이 3보다 큰지 여부를 지정한다. 마지막 정규(예를 들어, 컨텍스트) 코딩 계수의 위치는 변수 firstPosModel에 저장될 수 있다.
- [0029] 패스 2(a)에서, 계수 그룹의 제1 스캔 위치로부터 시작하여 제1PosModel 위치까지의 나머지 절대 레벨(예를 들어, abs\_remainder)의 코딩이 처리된다. 1보다 큰 플래그가 1과 동일한 위치만이 코딩된다. 비-이진 선택스 요소는 골롬-라이스 코드로 이진화되고, 결과적인 빈은 산술 코딩 엔진의 바이패스 모드로 코딩된다. 예를 들어, 2의 값은 "001"의 골롬-라이스 코드를 사용하여 나타낼 수 있고, "001"의 각 비트는 빈(예를 들어, bin 0, bin 0 및 bin 1)으로 지칭될 수 있다.
- [0030] 패스 2(b)에서, 절대 레벨(dec\_abs\_level)의 코딩은 제1 바이패스 코딩 위치(예를 들어, firstPosModel - 1, 역순으로)부터 시작하여 계수 그룹의 마지막 스캔 위치까지 처리되고, 골롬-라이스 코드를 사용하여 산술 코딩 엔진의 바이패스 모드로 완전히 코딩된다.
- [0031] 패스 3에서 부호 플래그(예를 들어, sign\_flag)의 코딩은 sig\_coeff\_flag가 1과 동일한 모든 스캔 위치에 대해 처리된다.
- [0032] 32개를 넘지 않는 정규-코딩된 빈(예를 들어, sig\_coeff\_flag, gt1\_flag, par\_level\_flag 및 gt3\_flag)이 4x4 서브-블록에 대해 코딩되거나 또는 디코딩되는 것이 예상될 수 있다. 2x2 크로마 서브-블록에 대해, 정규-코딩된 빈의 수는 8로 제한될 수 있다. 제한에 도달한 이후에, 모든 빈은 바이패스 모드로 코딩된다.
- [0033] 변환-스킵 잔차 블록에 대해 JVET에 의해 채택된 새로운 잔차 코딩 프로세스에서, 변환-스킵 잔차 코딩의 계수 스캐닝 순서는 순방향 스캐닝이며, 변환-스킵 블록의 좌측-상단 위치로부터 시작한다. 서브-블록의 변환-스킵 계수 레벨은 스캔 위치에 대해 여섯 개의 패스로 코딩된다.
- [0034] 패스 1에서, sig\_coeff\_flag, coeff\_sign\_flag, 1보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[0]), par\_level\_flag는 코딩 순서로 처리된다. sig\_coeff\_flag가 1과 동일한 경우, coeff\_sign\_flag 및 abs\_level\_gtx\_flag[0]는 순서대로 코딩된다. coeff\_sign\_flag는 변환 계수 레벨의 부호를 지정한다. abs\_level\_gtx\_flag[0]은 절대 레벨이 1보다 큰지 여부를 지정한다. abs\_level\_gtx\_flag[0]이 1과 동일한 경우, par\_level\_flag는 추가적으로 코딩된다. par\_level\_flag는 절대 레벨에서 2를 뺀 패리티를 지정한다. 임의의 플래그를 코딩하기 전에, 컨텍스트-적응형 이진 산술 코딩(context-adaptive binary arithmetic coding, "CABAC") 엔진은 컨텍스트 코딩된 빈을 이용 가능한지 여부를 검사한다. 컨텍스트 코딩된 빈이 이용 가능하지 않은 경우, 그 플래그는 바이패스 코딩된다.
- [0035] 패스 2에서, 주어진 위치에 대한 abs\_level\_gtx\_flag[0]이 1과 동일하면, 3보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[1])가 코딩된다. abs\_level\_gtx\_flag[1]은 절대 레벨이 3보다 큰지 여부를 지정한다. 각 계수의 abs\_level\_gtx\_flag[1]을 코딩하기 전에, CABAC 엔진은 컨텍스트 코딩된 빈이 이용 가능한지를 검사한다. 컨텍스트 코딩된 빈이 이용 가능하지 않은 경우, abs\_level\_gtx\_flag[1]은 바이패스 코딩된다.
- [0036] 패스 3에서, 주어진 위치에 대한 abs\_level\_gtx\_flag[1]이 1과 동일하면, 5보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[2])가 코딩된다. abs\_level\_gtx\_flag[2]는 절대 레벨이 5보다 큰지 여부를 지정한다. 각 계수의 abs\_level\_gtx\_flag[2]를 코딩하기 전에, CABAC 엔진은 컨텍스트 코딩된 빈이 이용 가능한지를 확인한다. 컨텍스트 코딩된 빈이 이용 가능하지 않은 경우, abs\_level\_gtx\_flag[2]는 바이패스 코딩된다.
- [0037] 패스 4에서, 주어진 위치에 대한 abs\_level\_gtx\_flag[2]가 1과 동일하면, 7보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[3])가 코딩된다. abs\_level\_gtx\_flag[3]은 절대 레벨이 7보다 큰지 여부를 지정한다. 각 계수의 abs\_level\_gtx\_flag[3]을 코딩하기 전에, CABAC 엔진은 컨텍스트 코딩된 빈이 이용 가능한지를 검사한다. 컨텍스트 코딩된 빈이 이용 가능하지 않은 경우, abs\_level\_gtx\_flag[3]은 바이패스 코딩된다.
- [0038] 패스 5에서, 위치에 대한 abs\_level\_gtx\_flag[3]이 1과 동일한 경우, 9보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[4])가 코딩된다. abs\_level\_gtx\_flag[4]는 절대 레벨이 9보다 큰지 여부를 지정한다. 각 계수의 abs\_level\_gtx\_flag[4]를 코딩하기 전에, CABAC 엔진은 컨텍스트 코딩된 빈이 이용 가능한지를

검사한다. 컨텍스트 코딩된 빈이 이용 가능하지 않은 경우, `abs_level_gtx_flag[4]`는 바이패스 코딩된다.

- [0039] 패스 6에서, `abs_remainder`는 `abs_level_gtx_flag[4]`가 1과 동일한 모든 스캔 위치에 대해 처리된다. 비-이진 선택스 요소는 곱셈-라이트 코드로 이진화되고, 결과적인 빈은 산술 코딩 엔진의 바이패스 모드로 코딩된다.
- [0040] 도 3은 코딩을 변환하기 위한 선택스를 포함하는 예시적인 의사코드를 예시한다. 예를 들어, 도 3에 도시된 선택스는 VVC에서 코딩을 변환하는 데 사용될 수 있다. 도 4는 스킵 잔차 코딩을 변환하는 선택스를 포함하는 예시적인 의사코드를 예시한다. 예를 들어, 도 4에 도시된 선택스는 VVC에서 스킵 잔차 코딩을 변환하는 데 사용될 수 있다.
- [0041] 변환-스킵 잔차 코딩의 현재 설계에는 여러 문제가 있다. 첫째, 변환-스킵 잔차 코딩의 코딩 패스의 수는 6이다. 즉, 다수의 경우에서, CABAC 엔진은 계수 그룹을 여섯 번 스캔해야 하며, 이는 CABAC의 처리량에 상당한 영향을 미친다. 둘째, 변환-스킵 잔차 코딩의 코딩 패스의 수는 변환 잔차 코딩의 것과 상이하다(예를 들어, 여섯 개의 패스 대 세 개의 패스). 코딩 패스 수에서의 차이는 하드웨어 구현 복잡성을 초래할 수 있다. 셋째, 변환-스킵 잔차 코딩의 계수 스캐닝은 순방향 스캔인 한편, 변환된 잔차 코딩의 스캐닝은 역순이다. 스캐닝 순서에서의 차이는 또한, 하드웨어 구현 복잡성을 초래할 수 있다. 넷째, 변환된 잔차 코딩에서, 바이패스 코딩은 두 개의 선택스 요소(예를 들어, `abs_remainder` 및 `dec_abs_level`)만을 갖는다. 하지만, 변환-스킵 잔차 코딩에서, 바이패스 코딩은 `sig_coeff_flag`, `coeff_sign_flag`, `par_level_flag`, `abs_level_gtx_flag[0]`, `abs_level_gtx_flag[1]`, `abs_level_gtx_flag[2]`, `abs_level_gtx_flag[3]`, `abs_level_gtx_flag[4]` 및 `abs_remainder`와 같은 다수의 더 많은 선택스 요소를 가질 수 있다. 변환된 잔차 코딩 및 변환-스킵 잔차 코딩의 바이패스 코딩 방식을 통합하는 것이 바람직하다.
- [0042] 본 개시의 실시예는 패리티 플래그의 새로운 바이패스 코딩 방법을 제공한다. 패리티 플래그는 0이 아닌 계수 위치의 절대 값이 짝수인지 홀수인지를 나타내는, 절대 레벨에서 2를 뺀 패리티를 지정한다. 변환-스킵 블록에서 패리티 플래그의 확률 분포는 균일할 수 있다. 그러므로, 짝수 값 및 홀수 값의 확률은 동일할 수 있다. 결과로서, 패리티 플래그 및 신호 패리티 플래그의 바이패스 코딩이 적용될 수 있다. 이 바이패스 코딩은 `abs_remainder`가 시그널링되기 전 또는 그 이후에 이루어질 수 있다. 1보다 큰 플래그가 1과 동일한 경우, 패리티 플래그가 시그널링될 수 있다. 일부 실시예에서, 컨텍스트 코딩된 빈의 총 수가 변환 블록에서 샘플당 2개의 빈으로 제한될 수 있기 때문에, 패리티 플래그의 컨텍스트를 해제하는 것(freeing up)은 동일하지 않은 확률 분포(예를 들어, 패리티 플래그 이후에 코딩된 x보다 더 큰 플래그 중 임의의 것)를 갖는 다른 선택스 요소가 컨텍스트 코딩되는 것을 허용할 수 있고, 이는 코딩 효율성을 개선할 수 있다.
- [0043] 일부 실시예에서, 패리티 플래그는 정규 컨텍스트 코딩된 빈으로서 코딩되고, 계수 코딩 프로세스의 제2 패스에서 시그널링될 수 있다. 패리티 플래그는 3보다 큰 플래그 이전 또는 그 이후에 시그널링될 수 있다.
- [0044] VVC(예를 들어, VVC 5)에서, 총 네 개의 코딩 패스가 3보다 큰 플래그(예를 들어, `abs_level_gtx_flag[1]`), 5보다 큰 플래그(예를 들어, `abs_level_gtx_flag[2]`), 7보다 큰 플래그(`abs_level_gtx_flag[3]`) 및 9보다 큰 플래그(예를 들어, `abs_level_gtx_flag[4]`)를 코딩하는 데 사용된다. 즉, 각 플래그는 별도의 코딩 패스에서 코딩된다. 본 개시의 실시예는 단일 패스에서 모든 플래그를 코딩하기 위한 새로운 방법을 제공한다. 결과적으로, 모든 플래그를 코딩하기 위해 하나의 코딩 패스만이 필요하며, 이는 CABAC의 처리량을 개선할 수 있다.
- [0045] 일부 실시예에서, 3보다 큰 플래그(예를 들어, `abs_level_gtx_flag[1]`)의 위치는 제1 코딩 패스로 이동될 수 있다. 예를 들어, 3보다 큰 플래그는 1보다 큰 플래그(예를 들어, `abs_level_gtx_flag[1]`)가 1과 동일한 경우 시그널링될 수 있다. 이 변경은 1보다 큰 플래그 및 3보다 큰 플래그의 코딩을 변환된 잔차 코딩 케이스와 통합할 수 있으며, 이는 또한 제1 코딩 패스에서 이 두 개의 플래그를 코딩한다.
- [0046] 일부 실시예에서, 서브-블록의 변환-스킵 레벨의 코딩 패스의 수는 3으로 감소될 수 있다. 도 5는 본 개시의 일부 실시예에 따른, 코딩 패스의 수가 3으로 감소된 변환-스킵 잔차 코딩의 예시적인 방법을 예시한다. 도 5에서의 방법은 세 개의 패스를 포함한다.
- [0047] 패스 1(단계 502)에서, 서브-블록의 계수가 스캔된다. 일부 실시예에서, 서브-블록의 제1 스캔 위치로부터 시작하여 그 서브-블록의 마지막 스캔 위치까지의 각 계수가 스캔된다. 각 계수에 대해 중요도 플래그(예를 들어, `sig_coeff_flag`)가 있을 수 있다. 일부 실시예에서, 중요도 플래그는 계수의 레벨이 0이 아닌 값인지를 지정할 수 있다. 계수에 대한 중요도 플래그가 레벨이 0이 아닌 값(예를 들어, 중요도 플래그가 1과 동일함)임을 나타내는 경우, 신호 계수 부호 플래그(예를 들어, `coeff_sign_flag`) 및 1보다 큰 플래그(예를 들어, `abs_level_gtx_flag[0]`)가 코딩될 수 있다. 1보다 큰 플래그는 레벨의 절대 값이 1보다 큰지 여부를 지정할 수

있다.

- [0048] 패스 2(단계 504)에서, 서브-블록의 계수가 스캔된다. 일부 실시예에서, 서브-블록의 제1 스캔 위치로부터 시작하여 그 서브-블록의 마지막 스캔 위치까지의 각 계수가 스캔된다. 계수에 대한 1보다 큰 플래그가 절대 레벨이 1보다 크다는 것을 나타내는 경우(예를 들어, 1보다 큰 플래그가 1과 동일함), 3보다 큰 플래그(예를 들어, `abs_level_gtx_flag[1]`)가 코딩될 수 있다. 3보다 큰 플래그는 절대 레벨이 3보다 큰지 여부를 지정할 수 있다. 계수에 대한 3보다 큰 플래그가 절대 레벨이 3보다 크다는 것을 나타내는 경우(예를 들어, 3보다 큰 플래그가 1과 동일함), 5보다 큰 플래그(예를 들어, `abs_level_gtx_flag[2]`)가 코딩될 수 있다. 5보다 큰 플래그는 절대 레벨이 5보다 큰지 여부를 지정할 수 있다. 계수에 대한 5보다 큰 플래그가 절대 레벨이 5보다 크다는 것을 나타내는 경우(예를 들어, 5보다 큰 플래그가 1과 동일함), 7보다 큰 플래그(예를 들어, `abs_level_gtx_flag[3]`)가 코딩될 수 있다. 7보다 큰 플래그는 절대 레벨이 7보다 큰지 여부를 지정할 수 있다. 계수에 대한 7보다 큰 플래그가 절대 레벨이 7보다 크다는 것을 나타내는 경우(예를 들어, 7보다 큰 플래그가 1과 동일함), 9보다 큰 플래그(예를 들어, `abs_level_gtx_flag[4]`)가 코딩될 수 있다. 9보다 큰 플래그는 절대 레벨이 9보다 큰지 여부를 지정할 수 있다.
- [0049] 패스 3(단계 506)에서, 서브-블록의 계수가 스캔된다. 일부 실시예에서, 서브-블록의 제1 스캔 위치로부터 시작하여 그 서브-블록의 마지막 스캔 위치까지의 각 계수가 스캔된다. 계수에 대한 1보다 큰 플래그가 절대 레벨이 1보다 크다는 것을 나타내는 경우(예를 들어, 1보다 큰 플래그가 1과 동일함), 계수에 대한 패리티 레벨 플래그(예를 들어, `par_level_flag`)는 바이패스 코딩될 수 있다. 패리티 레벨 플래그는 절대 레벨에서 2를 뺀 패리티를 지정할 수 있다. 9보다 큰 플래그가 절대 레벨이 9보다 크다는 것을 나타내는 경우(예를 들어, 9보다 큰 플래그가 1과 동일함), 계수에 대한 나머지 절대 레벨(예를 들어, `abs_remainder`)이 코딩될 수 있고, 비-이진 선택스 요소는 콜롬-라이스 코드로 이진화될 수 있다. 일부 실시예에서, 결과적인 빈은 산술 코딩 엔진의 바이패스 모드로 코딩될 수 있다.
- [0050] 도 6은 본 개시의 일부 실시예에 따른 도 5에 도시된 방법의 선택스를 포함하는 예시적인 의사코드를 예시한다. 도 6에서의 의사코드의 일부는 이탤릭체로 되어 있고, 3보다 큰 플래그, 5보다 큰 플래그, 7보다 큰 플래그, 9보다 큰 플래그 및 패리티 레벨 플래그의 처리를 나타낸다.
- [0051] 도 5의 방법이 인코더(예를 들어, 도 1의 인코더)에 의해 구현될 수 있다는 것이 인식된다. 일부 실시예에서, 인코더는 비디오 프레임을 수신할 수 있다. 디코더(예를 들어, 도 2의 디코더)에 대해, 방법은 세 개의 패스를 포함할 수 있다.
- [0052] 패스 1에서, 서브-블록의 계수가 스캔된다. 일부 실시예에서, 서브-블록의 제1 스캔 위치로부터 시작하여 그 서브-블록의 마지막 스캔 위치까지의 각 계수가 스캔된다. 각 계수에 대해 중요도 플래그(예를 들어, `sig_coeff_flag`)가 있을 수 있다. 중요도 플래그가 디코딩될 수 있다. 중요도 플래그는 레벨이 0이 아닌 값인지를 지정할 수 있다. 계수에 대한 중요도 플래그가 레벨이 0이 아닌 값(예를 들어, 중요도 플래그가 1과 동일함)임을 나타내는 경우, 신호 계수 부호 플래그(예를 들어, `coeff_sign_flag`) 및 1보다 큰 플래그(예를 들어, `abs_level_gtx_flag[0]`)가 디코딩될 수 있다.
- [0053] 패스 2에서, 서브-블록의 계수가 스캔된다. 일부 실시예에서, 서브-블록의 제1 스캔 위치로부터 시작하여 그 서브-블록의 마지막 스캔 위치까지의 각 계수가 스캔된다. 계수에 대한 1보다 큰 플래그가 절대 레벨이 1보다 크다는 것을 나타내는 경우(예를 들어, 1보다 큰 플래그가 1과 동일함), 각 계수에 대한 패리티 레벨 플래그(예를 들어, `par_level_flag`)는 디코딩될 수 있다. 패리티 레벨 플래그는 절대 레벨에서 2를 뺀 패리티를 지정할 수 있다. 계수에 대한 1보다 큰 플래그가 절대 레벨이 1보다 크다는 것을 나타내는 경우(예를 들어, 1보다 큰 플래그가 1과 동일함), 3보다 큰 플래그(예를 들어, `abs_level_gtx_flag[1]`)가 디코딩될 수 있다. 계수에 대한 3보다 큰 플래그가 절대 레벨이 3보다 크다는 것을 나타내는 경우(예를 들어, 3보다 큰 플래그가 1과 동일함), 5보다 큰 플래그(예를 들어, `abs_level_gtx_flag[2]`)가 디코딩될 수 있다. 계수에 대한 5보다 큰 플래그가 절대 레벨이 5보다 크다는 것을 나타내는 경우(예를 들어, 4보다 큰 플래그가 1과 동일함), 7보다 큰 플래그(예를 들어, `abs_level_gtx_flag[3]`)가 디코딩될 수 있다. 계수에 대한 7보다 큰 플래그가 절대 레벨이 7보다 크다는 것을 나타내는 경우(예를 들어, 7보다 큰 플래그가 1과 동일함), 9보다 큰 플래그(예를 들어, `abs_level_gtx_flag[4]`)가 디코딩될 수 있다.
- [0054] 패스 3에서, 서브-블록의 계수가 스캔된다. 일부 실시예에서, 서브-블록의 제1 스캔 위치로부터 시작하여 그 서브-블록의 마지막 스캔 위치까지의 각 계수가 스캔된다. 9보다 큰 플래그가 절대 레벨이 9보다 크다는 것을 나타내는 경우(예를 들어, 9보다 큰 플래그가 1과 동일함), 계수에 대한 나머지 절대 레벨(예를 들어,

abs\_remainder)이 디코딩될 수 있다.

- [0055] 일부 실시예에서, 디코더는 비디오 비트-스트림을 수신할 수 있다.
- [0056] 도 7은 본 개시의 일부 실시예에 따른, 코딩 패스의 수가 3으로 감소된 변환-스킵 잔차 코딩의 다른 예시적인 방법을 예시한다. 도 7의 방법은 세 개의 패스를 포함할 수 있다.
- [0057] 패스 1(단계 702)에서, 서브-블록의 계수가 스캔된다. 일부 실시예에서, 서브-블록의 제1 스캔 위치로부터 시작하여 그 서브-블록의 마지막 스캔 위치까지의 각 계수가 스캔된다. 각 계수에 대해 중요도 플래그(예를 들어, sig\_coeff\_flag)가 있을 수 있다. 중요도 플래그는 레벨이 0이 아닌 값인지를 나타낼 수 있다. 계수에 대한 중요도 플래그가 레벨이 0이 아닌 값(예를 들어, 중요도 플래그가 1과 동일함)임을 나타내는 경우, 신호 계수 부호 플래그(예를 들어, coeff\_sign\_flag) 및 1보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[0])가 코딩될 수 있다. 1보다 큰 플래그는 레벨의 절대 값이 1보다 큰지 여부를 지정할 수 있다. 계수에 대한 1보다 큰 플래그가 레벨의 절대 값이 1보다 크다는 것을 나타내는 경우(예를 들어, 1보다 큰 플래그가 1과 동일함), 3보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[1])가 코딩될 수 있다. 3보다 큰 플래그는 절대 레벨이 3보다 큰지 여부를 지정할 수 있다.
- [0058] 패스 2(단계 704)에서, 서브-블록의 계수가 스캔된다. 일부 실시예에서, 서브-블록의 제1 스캔 위치로부터 시작하여 그 서브-블록의 마지막 스캔 위치까지의 각 계수가 스캔된다. 계수에 대한 3보다 큰 플래그가 절대 레벨이 3보다 크다는 것을 나타내는 경우(예를 들어, 3보다 큰 플래그가 1과 동일함), 5보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[2])가 코딩될 수 있다. 5보다 큰 플래그는 절대 레벨이 5보다 큰지 여부를 지정할 수 있다. 계수에 대한 5보다 큰 플래그가 절대 레벨이 5보다 크다는 것을 나타내는 경우(예를 들어, 5보다 큰 플래그가 1과 동일함), 7보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[3])가 코딩될 수 있다. 7보다 큰 플래그는 절대 레벨이 7보다 큰지 여부를 지정할 수 있다. 계수에 대한 7보다 큰 플래그가 절대 레벨이 7보다 크다는 것을 나타내는 경우(예를 들어, 7보다 큰 플래그가 1과 동일함), 9보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[4])가 코딩될 수 있다. 9보다 큰 플래그는 절대 레벨이 9보다 큰지 여부를 지정할 수 있다.
- [0059] 패스 3(단계 706)에서, 서브-블록의 계수가 스캔된다. 일부 실시예에서, 서브-블록의 제1 스캔 위치로부터 시작하여 그 서브-블록의 마지막 스캔 위치까지의 각 계수가 스캔된다. 계수에 대한 1보다 큰 플래그가 절대 레벨이 1보다 크다는 것을 나타내는 경우(예를 들어, 1보다 큰 플래그가 1과 동일함), 계수에 대한 패리티 레벨 플래그(예를 들어, par\_level\_flag)는 바이패스 코딩될 수 있다. 패리티 레벨 플래그는 절대 레벨에서 2를 뺀 패리티를 지정할 수 있다. 9보다 큰 플래그가 절대 레벨이 9보다 크다는 것을 나타내는 경우(예를 들어, 9보다 큰 플래그가 1과 동일함), 계수에 대한 나머지 절대 레벨(예를 들어, abs\_remainder)이 처리될 수 있고, 비-이진 선택스 요소는 콜롬-라이스 코드로 이진화될 수 있다. 결과적인 빈은 산술 코딩 엔진의 바이패스 모드로 코딩될 수 있다.
- [0060] 도 5에서의 방법과 비교하여, 도 7에서의 방법은 패스 2 대신 패스 1에서 3보다 큰 플래그의 코딩을 처리한다. 도 8은 본 개시의 일부 실시예에 따른, 도 7에 도시된 방법의 선택스를 포함하는 예시적인 의사코드를 예시한다. 도 8에서의 의사코드의 일부는 이텔릭체로 되어 있고, 3보다 큰 플래그, 5보다 큰 플래그, 7보다 큰 플래그, 9보다 큰 플래그 및 패리티 레벨 플래그의 처리를 나타낸다.
- [0061] 도 7에서의 방법은 인코더(예를 들어, 도 1의 인코더)에 의해 구현될 수 있다는 것이 인식된다. 일부 실시예에서, 인코더는 비디오 프레임을 수신할 수 있다. 디코더(예를 들어, 도 2의 디코더)에 대해, 방법은 세 개의 패스를 포함할 수 있다.
- [0062] 패스 1에서, 서브-블록의 계수가 스캔된다. 일부 실시예에서, 서브-블록의 제1 스캔 위치로부터 시작하여 그 서브-블록의 마지막 스캔 위치까지의 각 계수가 스캔된다. 각 계수에 대해 중요도 플래그(예를 들어, sig\_coeff\_flag)가 있을 수 있다. 중요도 플래그가 디코딩된다. 중요도 플래그는 레벨이 0이 아닌 값인지를 나타낼 수 있다. 계수에 대한 중요도 플래그가 레벨이 0이 아닌 값(예를 들어, 중요도 플래그가 1과 동일함)임을 나타내는 경우, 신호 계수 부호 플래그(예를 들어, coeff\_sign\_flag) 및 1보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[0])가 디코딩될 수 있다. 계수에 대한 1보다 큰 플래그가 절대 레벨이 1보다 크다는 것을 나타내는 경우(예를 들어, 1보다 큰 플래그가 1과 동일함), 3보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[1])가 디코딩될 수 있다.
- [0063] 패스 2에서, 서브-블록의 계수가 스캔된다. 일부 실시예에서, 서브-블록의 제1 스캔 위치로부터 시작하여 그 서브-블록의 마지막 스캔 위치까지의 각 계수가 스캔된다. 계수에 대한 1보다 큰 플래그가 절대 레벨이 1보다 크

다는 것을 나타내는 경우(예를 들어, 1보다 큰 플래그가 1과 동일함), 각 계수에 대한 패리티 레벨 플래그(예를 들어, `par_level_flag`)는 디코딩될 수 있다. 계수에 대한 3보다 큰 플래그가 절대 레벨이 3보다 크다는 것을 나타내는 경우(예를 들어, 3보다 큰 플래그가 1과 동일함), 5보다 큰 플래그(예를 들어, `abs_level_gtx_flag[2]`)가 디코딩될 수 있다. 계수에 대한 5보다 큰 플래그가 절대 레벨이 5보다 크다는 것을 나타내는 경우(예를 들어, 5보다 큰 플래그가 1과 동일함), 7보다 큰 플래그(예를 들어, `abs_level_gtx_flag[3]`)가 디코딩될 수 있다. 계수에 대한 7보다 큰 플래그가 절대 레벨이 7보다 크다는 것을 나타내는 경우(예를 들어, 7보다 큰 플래그가 1과 동일함), 9보다 큰 플래그(예를 들어, `abs_level_gtx_flag[4]`)가 디코딩될 수 있다. 9보다 큰 플래그는 절대 레벨이 9보다 큰지 여부를 지정할 수 있다.

[0064] 패스 3에서, 서브-블록의 계수가 스캔된다. 일부 실시예에서, 서브-블록의 제1 스캔 위치로부터 시작하여 그 서브-블록의 마지막 스캔 위치까지의 각 계수가 스캔된다. 9보다 큰 플래그가 절대 레벨이 9보다 크다는 것을 나타내는 경우(예를 들어, 9보다 큰 플래그가 1과 동일함), 계수에 대한 나머지 절대 레벨(예를 들어, `abs_remainder`)이 디코딩될 수 있다.

[0065] 일부 실시예에서, 디코더는 비디오 비트-스트림을 수신할 수 있다.

[0066] 일부 실시예에서, 변환-스킵 잔차 블록의 스캐닝 순서는 변환된 및 변환-스킵 잔차 코딩 프로세스의 스캐닝 순서를 통합하기 위해 변경될 수 있다. 예를 들어, 변환-스킵 잔차 블록의 스캐닝 순서는 순방향 스캐닝으로부터 역방향 스캐닝으로 변경될 수 있다. 도 9는 본 개시의 일부 실시예에 따른, 8x8 변환-스킵 블록의 예시적인 역방향 스캐닝을 예시한다. 도 9에 도시된 바와 같이, 계수의 스캐닝은 우측-하단 코너에 위치한 계수로부터 시작하여 좌측-상단 코너에 위치한 계수에서 끝난다. 도 9에 예시된 스캐닝 순서는 도 5 및 도 7에 도시된 방법의 의해 적용될 수 있다는 것이 인식된다.

[0067] 일부 실시예에서, 역방향 스캐닝은 변환 블록이 플립된 이후에 실행될 수 있다. 도 10a 및 도 10b는 본 개시의 일부 실시예에 따른, 8x8 블록의 예시적인 플립핑을 예시한다. 도 10a 및 도 10b에 도시된 바와 같이, 블록을 플립한 이후에, 좌측-상단 잔차 계수의 위치가 우측-하단 위치로 이동된다. 블록이 플립된 이후에, 역방향 스캐닝(예를 들어, 도 9의 역방향 스캐닝)이 실행될 수 있다.

[0068] 일부 실시예에서, 바이패스 코딩은 다수의 패스로 실행될 수 있다. 도 11은 본 개시의 일부 실시예에 따른, 예시적인 다중-패스 코딩을 예시한다. 일부 실시예에서, 도 11에 도시된 다중-패스 코딩은 VVC(예를 들어, VVC 5)에서 실행될 수 있다. 도 11에서의 컨텍스트 코딩된 bin의 수가 제1 패스의 위치에서 최대 제한에 도달했다고 (도 11에서 검은 점으로 표시됨) 간주될 수 있다. 도 11에 도시된 바와 같이, 플래그(예를 들어, 3보다 큰 플래그, 5보다 큰 플래그, 7보다 큰 플래그, 9보다 큰 플래그 등)는 다수의 코딩 패스에서 바이패스 코딩될 수 있다.

[0069] 일부 실시예에서, 레벨의 절대 값에 대한 단일 패스 바이패스 코딩이 달성될 수 있다. 도 12는 본 개시의 일부 실시예에 따른, 레벨의 절대 값에 대한 예시적인 단일 패스 바이패스 코딩 방법을 예시한다. 도 12에 도시된 바와 같이, 컨텍스트 코딩된 bin이 최대 제한에 도달하면(예를 들어, 도 12에서 흑색 점으로 표시됨), CABAC 엔진은 콜롬-rais 코딩을 사용하여 절대 레벨의 나머지 부분의 바이패스 코딩을 시작할 수 있다.

[0070] 도 13은 본 개시의 일부 실시예에 따른, 라이스 파라미터에 대한 예시적인 룩업 테이블을 예시한다. 도 13에 따라, 라이스 파라미터는 변수 `cRiceParam`으로 나타날 수 있다. 절대치 합계의 위치는 변수 `locSumAbs`에 의해 나타날 수 있다.

[0071] 일부 실시예에서, 라이스 파라미터(예를 들어, `cRiceParam`)는 다음 방식으로 도출될 수 있다. 변환-스킵 블록에 대한 어레이 `AbsLevel[x][y]`, 좌측-상단 루마 위치( $x_0, y_0$ ) 및 현재 계수 스캔 위치( $x_C, y_C$ )가 주어지면, `minLevel`이 최소 바이패스 코딩된 값이라고 간주하고, 계수의 플래그 중 어느 것도 컨텍스트 코딩되지 않은 경우, 그 계수에 대한 `minLevel`은 0이다. 플래그의 모두가 컨텍스트 코딩된 경우, `minLevel`은 10과 동일하다.

[0072] 일부 실시예에서, 변수 `locSumAbs`는 다음 의사코드에 의해 지정된 바와 같이 도출될 수 있다.

[0073] `locSumAbs = 0`

[0074] `if (xC > 0)`

[0075] `locSumAbs += AbsLevel[ xC - 1][ yC ]`

[0076] `if (yC > 0)`

- [0077] `locSumAbs += AbsLevel[ xC ] [ yC -1]`
- [0078] `locSumAbs += ( 10 - minLevel)`
- [0079] `locSumAbs = Clip3 ( 0, 31, LocSumAbs )`
- [0080] 일부 실시예에서, 단일 패스 바이패스 코딩은 도 7에 도시된 방법과 결합될 수 있다. 예를 들어, 도 14는 본 개시의 일부 실시예에 따른, 단일 패스 바이패스 코딩과 결합된 코딩 패스의 수가 3으로 감소된 변환-스킵 잔차 코딩의 예시적인 방법을 예시한다. 도 14에서의 방법은 3개의 패스를 포함한다.
- [0081] 패스 1(단계 1402)에서, 서브-블록의 계수가 스캔된다. 일부 실시예에서, 서브-블록의 제1 스캔 위치로부터 시작하여 마지막 스캔 위치까지의 각 계수가 스캔된다. 각 계수에 대해, 컨텍스트 코딩된 bin의 나머지 수가 그룹 제한 이상인 경우 다음이 수행될 수 있다. 예를 들어, 일부 실시예에서, 컨텍스트 코딩 또는 바이패스 코딩을 사용하여 그룹에서의 플래그의 모두를 코딩하는 것이 더 효율적이다. 결과로서, 컨텍스트 코딩된 bin의 수가 그룹 제한보다 적은 경우, 컨텍스트 코딩을 사용하여 나머지 플래그 중 일부를 코딩하고 바이패스 코딩을 사용하여 나머지 플래그를 코딩하는 것 대신에, 바이패스 코딩을 사용하여 나머지 플래그를 모두 코딩하는 것이 더 효율적이다. 이 예시에서, 패스 1에서의 그룹에서 코딩되는 4개의 플래그(예를 들어, 중요도 플래그, 신호 계수 부호 플래그, 1보다 큰 플래그 및 3보다 큰 플래그)가 있을 수 있다. 그러므로, 그룹 제한이 4로 주어지면, 컨텍스트 코딩된 bin의 나머지 수가 4 이상인 경우 각 계수에 대해 다음이 수행될 수 있다. 중요도 플래그(예를 들어, `sig_coeff_flag`)가 레벨이 0이 아닌 값(예를 들어, 중요도 플래그가 1과 동일함)임을 나타내는 경우, 신호 계수 부호 플래그(예를 들어, `coeff_sign_flag`) 및 1보다 큰 플래그(예를 들어, `abs_level_gtx_flag[0]`)가 코딩될 수 있다. 1보다 큰 플래그가 절대 레벨이 1보다 크다는 것을 나타내는 경우(예를 들어, 1보다 큰 플래그가 1과 동일함), 3보다 큰 플래그(예를 들어, `abs_level_gtx_flag[1]`)가 코딩될 수 있다. 일부 실시예에서, 컨텍스트 코딩된 bin이 최대 제한(예를 들어, 도 12의 검은 점)에 도달할 때까지 계수가 스캔될 수 있고, 그 후 스캔은 제1 패스의 마지막 위치로서의 위치에서 중지된다.
- [0082] 패스 1의 종료 이후에 및 패스 2의 시작 이전에, 제1 패스 바이패스 위치 변수(예를 들어, `iFirstPassBypassPos`)는 이전 패스의 마지막 위치에 따라 설정될 수 있다(단계 1404). 일부 실시예에서, 제1 패스 바이패스 위치 변수는 이전 패스의 마지막 위치에 1을 더한 것으로 설정될 수 있다.
- [0083] 패스 2(단계 1406)에서, 서브-블록의 제1 스캔 위치로부터 시작하여 `iFirstPassBypassPos`까지의 계수가 스캔될 수 있다. 각 계수에 대해, 컨텍스트 코딩된 bin의 나머지 수가 그룹 제한 이상인 경우 다음이 수행될 수 있다. 예를 들어, 그룹 제한은 3일 수 있다(예를 들어, 5보다 큰 플래그, 7보다 큰 플래그 및 9보다 큰 플래그). 3보다 큰 플래그가 절대 레벨이 3보다 크다는 것을 나타내는 경우(예를 들어, 3보다 큰 플래그가 1과 동일함), 5보다 큰 플래그(예를 들어, `abs_level_gtx_flag[2]`)가 코딩될 수 있다. 5보다 큰 플래그가 절대 레벨이 5보다 크다는 것을 나타내는 경우(예를 들어, 5보다 큰 플래그가 1과 동일함), 7보다 큰 플래그(예를 들어, `abs_level_gtx_flag[3]`)가 코딩될 수 있다. 7보다 큰 플래그가 절대 레벨이 7보다 크다는 것을 나타내는 경우(예를 들어, 7보다 큰 플래그가 1과 동일함), 9보다 큰 플래그(예를 들어, `abs_level_gtx_flag[4]`)가 코딩될 수 있다.
- [0084] 패스 2의 종료 이후에 및 패스 3의 시작 이전에, 제2 패스 바이패스 위치 변수(예를 들어, `iSecondPassBypassPos`)는 이전 패스의 마지막 위치에 따라 설정된다(단계 1408). 일부 실시예에서, 제2 패스 바이패스 위치 변수는 이전 패스의 마지막 위치에 1을 더한 것으로 설정될 수 있다.
- [0085] 패스 3(a)(단계 1410)에서, 서브-블록의 제1 스캔 위치로부터 시작하여 `iFirstPassBypassPos`까지의 계수가 스캔될 수 있다. 각 계수에 대해 다음이 수행될 수 있다. 1보다 큰 플래그가 절대 레벨이 1보다 크다는 것을 나타내는 경우(예를 들어, 1보다 큰 플래그가 1과 동일함), 패리티 레벨 플래그(예를 들어, `par_level_flag`)는 바이패스 코딩될 수 있다. 9보다 큰 플래그가 절대 레벨이 9보다 크다는 것을 나타내는 경우(예를 들어, 9보다 큰 플래그가 1과 동일함), 나머지 절대 레벨(예를 들어, `abs_remainder`)의 코딩은 곱셈-라이트 코드로 이진화된 비-이진 선택스 요소에 의해 처리될 수 있다. 결과적인 bin은 산술 코딩 엔진의 바이패스 모드로 코딩될 수 있다.
- [0086] 패스 3(b)(단계 1412)에서, `iFirstPassBypassPos`로부터 시작하여 마지막 스캔 위치까지의 계수가 스캔될 수 있다. 각 계수에 대해, 절대 레벨(예를 들어, `dec_abs_level`)의 코딩은 곱셈-라이트 코드로 이진화된 비-이진 선택스 요소에 의해 처리될 수 있다. 결과적인 bin은 산술 코딩 엔진의 바이패스 모드로 코딩될 수 있다. 신호 계수 부호 플래그가 또한 코딩될 수 있다.

- [0087] 일부 실시예에서, 패스 3(b)에서, iFirstPassBypassPos로부터 시작하여 마지막 스캔 위치까지의 계수가 스캔될 수 있고, 절대 레벨(예를 들어, dec\_abs\_level)의 코딩은 각 계수에 대해 두 개의 단계에 의해 처리될 수 있다. 먼저, dec\_abs\_level이 0인지의 여부가 시그널링된다. dec\_abs\_level이 0이 아닌 경우 골롬-라이스 코드로 이진화된 비-이진 선택스 요소 및 결과적인 빈은 산술 코딩 엔진의 바이패스 모드로 코딩될 수 있다. 신호 계수 부호 플래그가 또한 코딩될 수 있다.
- [0088] 도 15는 본 개시의 일부 실시예에 따른, 도 14에서의 방법과 결합된 바이패스 코딩의 선택스를 포함하는 예시적인 의사코드를 예시한다. 도 15에서의 의사코드의 일부는 이탤릭체로 되어 있고, 3보다 큰 플래그, 5보다 큰 플래그, 7보다 큰 플래그, 9보다 큰 플래그 및 패리티 레벨 플래그의 처리를 나타낸다.
- [0089] 일부 실시예에서, 2-패스 코딩 방법이 사용될 수 있다. 예를 들어, 플래그의 모두는 패스 1에서 컨텍스트 코딩될 수 있고, 골롬-라이스 코딩은 패스 2에서 바이패스 코딩에 사용될 수 있다. 도 16은 본 개시의 일부 실시예에 따른, 컨텍스트 코딩을 위한 제1 패스 및 골롬-라이스 코딩(Golomb-Rice coding)을 위한 제2 패스를 갖는 변환-스킵 잔차 코딩의 예시적인 방법을 예시한다. 도 16에서의 방법은 2개의 패스를 포함한다.
- [0090] 패스 1(단계 1602)에서, 서브-블록의 계수가 스캔된다. 일부 실시예에서, 서브-블록의 제1 스캔 위치로부터 시작하여 마지막 스캔 위치까지의 각 계수가 스캔된다. 각 계수에 대해, 컨텍스트 코딩된 빈의 나머지 수가 8 이상인 경우, 다음이 수행될 수 있다. 중요도 플래그(예를 들어, sig\_coeff\_flag)는 컨텍스트 코딩될 수 있다. 중요도 플래그가 레벨이 0이 아님을 나타내는 경우(예를 들어, 중요도 플래그가 1과 동일함), 계수 부호 플래그(예를 들어, coeff\_sign\_flag) 및 1보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[0])는 시그널링될 수 있다. 1보다 큰 플래그가 절대 레벨이 1보다 크다는 것을 나타내는 경우(예를 들어, 1보다 큰 플래그가 1과 동일함), 패리티 레벨 플래그(예를 들어, par\_level\_flag) 및 3보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[1])가 코딩될 수 있다. 패리티 레벨 플래그는 절대 레벨에서 2를 뺀 패리티를 지정할 수 있다. 3보다 큰 플래그는 절대 레벨이 3보다 큰지 여부를 지정할 수 있다. 3보다 큰 플래그가 절대 레벨이 3보다 크다는 것을 나타내는 경우(예를 들어, 3보다 큰 플래그가 1과 동일함), 5보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[2])가 코딩될 수 있다. 5보다 큰 플래그가 절대 레벨이 5보다 크다는 것을 나타내는 경우(예를 들어, 5보다 큰 플래그가 1과 동일함), 7보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[3])가 코딩될 수 있다. 7보다 큰 플래그가 절대 레벨이 7보다 크다는 것을 나타내는 경우(예를 들어, 7보다 큰 플래그가 1과 동일함), 9보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[4])가 코딩될 수 있다. 9보다 큰 플래그는 레벨의 절대 레벨이 9보다 큰지 여부를 지정할 수 있다. 일부 실시예에서, 컨텍스트 코딩된 빈이 최대 제한(예를 들어, 도 12의 검은 점)에 도달할 때까지 계수가 스캔될 수 있고, 그 후 스캔은 제1 패스의 마지막 위치로서의 위치에서 중지된다.
- [0091] 패스 1의 종료 이후 및 패스 2의 시작 이전에, 이전 패스의 마지막 위치에 따라 제1 패스 바이패스 위치 변수(iFirstPassBypassPos)가 설정될 수 있다(단계 1604). 일부 실시예에서, 제1 패스 바이패스 위치 변수는 이전 패스의 마지막 위치에 1을 더한 값으로 설정될 수 있다. 제1 패스 바이패스 위치 변수는 절대 레벨(예를 들어, dec\_abs\_level) 선택스가 시그널링되는 시작 위치를 나타낼 수 있다. 스캔 위치가 제1 패스 바이패스 위치보다 작은 계수는 패스 1에서 컨텍스트 코딩을 통해 부분적으로 시그널링될 수 있고, 나머지 계수는 패스 2(a)에서 시그널링될 수 있다. 일부 실시예에서, 계수의 스캔 위치가 제1 패스 바이패스 위치 변수 이상인 경우, 그 계수의 플래그 중 어느 것도 컨텍스트 코딩되지 않으며, 전체 계수 및 부호는 패스 2(b)의 바이패스 코딩을 사용하여 시그널링될 수 있다.
- [0092] 패스 2(a)에서(단계 1606), 서브-블록의 제1 스캔 위치로부터 시작하여 제1 패스 바이패스 위치 변수에서 1을 뺀 위치까지의 계수가 스캔될 수 있다. 각 계수에 대해 다음이 수행될 수 있다. 9보다 큰 플래그가 절대 레벨이 9보다 크다는 것을 나타내는 경우(예를 들어, 9보다 큰 플래그가 1과 동일함), 나머지 절대 레벨(예를 들어, abs\_remainder)은 골롬-라이스 코드를 사용하여 이진화될 수 있고, 결과적인 빈은 산술 코딩 엔진의 바이패스 모드로 코딩될 수 있다.
- [0093] 패스 2(b)(단계 1608)에서, 제1 패스 바이패스 위치로부터 시작하여 마지막 스캔 위치까지 계수가 스캔될 수 있다. 각 계수에 대해 다음이 수행될 수 있다. 절대 레벨(예를 들어, dec\_abs\_level)은 골롬-라이스 코드로 이진화될 수 있으며, 결과적인 빈은 산술 코딩 엔진의 바이패스 모드로 코딩될 수 있다. 절대 레벨이 0과 동일하지 않은 경우, 계수 부호 플래그(예를 들어, coeff\_sign\_flag)는 바이패스 코딩될 수 있다.
- [0094] 도 16의 방법은 인코더(예를 들어, 도 1의 인코더)에 의해 구현될 수 있다는 것이 인식된다. 일부 실시예에서, 인코더는 비디오 프레임을 수신할 수 있다. 도 17의 방법으로부터 인코딩된 비디오 프레임은 디코더(예를 들어, 도 2의 디코더)를 사용하여 디코딩될 수 있다는 것이 인식된다. 일부 실시예에서, 디코딩 방법은 2개의 패스를

포함할 수 있다.

- [0095] 패스 1에서, 서브-블록의 계수가 스캔된다. 일부 실시예에서, 서브-블록의 제1 스캔 위치로부터 시작하여 마지막 스캔 위치까지의 각 계수가 스캔된다. 각 계수에 대해, 컨텍스트 코딩된 bin의 나머지 수가 그룹 제한 이상인 경우 다음이 수행될 수 있다. 예를 들어, 그룹 제한은 8일 수 있으며, 이는 패스 1에서 코딩될 상이한 플래그의 수(예를 들어, 중요도 플래그, 계수 부호 플래그, 1보다 큰 플래그, 패리티 레벨 플래그, 3보다 큰 플래그, 5보다 큰 플래그, 7보다 큰 플래그 및 9보다 큰 플래그)를 나타낸다. 중요도 플래그(예를 들어, sig\_coeff\_flag)는 컨텍스트 디코딩될 수 있다. 중요도 플래그가 레벨이 0이 아님을 나타내는 경우(예를 들어, 중요도 플래그가 1과 동일함), 계수 부호 플래그(예를 들어, coeff\_sign\_flag) 및 1보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[0])는 시그널링될 수 있다. 1보다 큰 플래그가 절대 레벨이 1보다 크다는 것을 나타내는 경우(예를 들어, 1보다 큰 플래그가 1과 동일함), 패리티 레벨 플래그(예를 들어, par\_level\_flag) 및 3보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[1])가 디코딩될 수 있다. 패리티 레벨 플래그는 절대 레벨에서 2를 뺀 패리티를 지정할 수 있다. 3보다 큰 플래그는 절대 레벨이 3보다 큰지 여부를 지정할 수 있다. 3보다 큰 플래그가 절대 레벨이 3보다 크다는 것을 나타내는 경우(예를 들어, 3보다 큰 플래그가 1과 동일함), 5보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[2])가 디코딩될 수 있다. 5보다 큰 플래그가 절대 레벨이 5보다 크다는 것을 나타내는 경우(예를 들어, 5보다 큰 플래그가 1과 동일함), 7보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[3])가 디코딩될 수 있다. 7보다 큰 플래그가 절대 레벨이 7보다 크다는 것을 나타내는 경우(예를 들어, 7보다 큰 플래그가 1과 동일함), 9보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[4])가 디코딩될 수 있다. 9보다 큰 플래그는 레벨의 절대 레벨이 9보다 큰지 여부를 지정할 수 있다. 일부 실시예에서, 컨텍스트 코딩된 bin이 최대 제한(예를 들어, 도 12의 검은 점)에 도달할 때까지 계수가 스캔될 수 있고, 그 후 스캔은 제1 패스의 마지막 위치로서의 위치에서 중지된다.
- [0096] 패스 1의 종료 이후 및 패스 2의 시작 이전에, 제1 패스 바이패스 위치 변수(iFirstPassBypassPos)는 이전 패스의 마지막 위치에 따라 설정될 수 있다. 일부 실시예에서, 제1 패스 바이패스 위치 변수는 이전 패스의 마지막 위치에 1을 더한 것으로 설정될 수 있다. 제1 패스 바이패스 위치 변수는 절대 레벨(예를 들어, dec\_abs\_level) 선택스가 시그널링되는 시작 위치를 나타낼 수 있다. 스캔 위치가 제1 패스 바이패스 위치보다 작은 계수는 패스 1에서 컨텍스트 코딩을 통해 부분적으로 시그널링될 수 있고, 나머지 계수는 패스 2(a)에서 시그널링될 수 있다. 일부 실시예에서, 계수의 스캔 위치가 제1 패스 바이패스 위치 변수 이상인 경우, 그 계수의 플래그 중 어느 것도 컨텍스트 디코딩되지 않고, 전체 계수 및 부호는 패스 2(b)의 바이패스 디코딩을 사용하여 시그널링될 수 있다.
- [0097] 패스 2(a)에서, 서브-블록의 제1 스캔 위치로부터 시작하여 제1 패스 바이패스 위치 변수에서 1을 뺀 위치까지의 계수가 스캔될 수 있다. 각 계수에 대해 다음이 수행될 수 있다. 9보다 큰 플래그가 절대 레벨이 9보다 크다는 것을 나타내는 경우(예를 들어, 9보다 큰 플래그는 1과 동일함), 이진화된 나머지 절대 레벨(예를 들어, abs\_remainder)은 콜롬-라이스 코드를 사용하여 디코딩될 수 있고, 결과적인 bin은 산술 코딩 엔진의 바이패스 모드로 디코딩될 수 있다.
- [0098] 패스 2(b)에서, 제1 패스 바이패스 위치로부터 시작하여 마지막 스캔 위치까지의 계수가 스캔될 수 있다. 각 계수에 대해 다음이 수행될 수 있다. 이진화된 절대 레벨(예를 들어, dec\_abs\_level)은 콜롬-라이스 코드로 디코딩될 수 있고, 결과적인 bin은 산술 코딩 엔진의 바이패스 모드로 디코딩될 수 있다. 절대 레벨이 0과 동일하지 않은 경우, 계수 부호 플래그(예를 들어, coeff\_sign\_flag)는 바이패스 디코딩될 수 있다.
- [0099] 도 17은 본 개시의 일부 실시예에 따른, 도 16에서의 방법의 바이패스 코딩의 선택스를 포함하는 예시적인 의사 코드를 예시한다. 도 17에서의 의사코드의 일부는 이탤릭체로 되어 있고, 이는 제1 패스 바이패스 위치 및 패스 2(b)의 처리를 나타낸다.
- [0100] 일부 실시예에서, 도 17에 도시된 바와 같이, 패스 1은 컨텍스트 코딩된 bin의 나머지 수가 8 이상인 경우에만(예를 들어, 도 17에 도시된 "MaxCcbs >=8") 수행된다. 이는 컨텍스트 코딩 bin 예산에서 최대 7개의 컨텍스트 코딩이 "낭비"될 수 있고, 이는 코딩 성능을 저하시킬 수 있다. 그러므로, 일부 실시예에서, abs\_level\_gtx\_flag[] 플래그의 수는 개시된 2-패스 코딩 방법에서 조정될 수 있다. 예를 들어, 9보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[4])를 코딩하는 대신에, 최대 7보다 큰 플래그(예를 들어, abs\_level\_gtx\_flag[3])만이 코딩된다. 따라서, 패스 1은 컨텍스트 코딩된 bin의 나머지 수가 7 이상인 경우에만 수행될 수 있다. 일부 실시예에서, 최대 5보다 큰 플래그(abs\_level\_gtx\_flag[2])만이 코딩될 수 있다. 따라서, 패스 1은 컨텍스트 코딩된 bin의 나머지 수가 6 이상인 경우에만 수행된다. abs\_level\_gtx\_flag[] 플래그의 수는

임의의 수로 조정될 수 있음이 인식된다. `abs_level_gtx_flag[]` 플래그의 수를 조정하는 것은, 제1 코딩 패스에서 더 많은 위치가 코딩되는 것을 허용할 수 있으며, 따라서 더 나은 코딩 효율을 제공할 수 있다.

[0101] 일부 실시예에서, 라이스 파라미터 `cRiceParam`은 다음 방식으로 도출될 수 있다. 스캔 위치(`xC`, `yC`)에서의 계수 값으로서 어레이 `TransCoeffLevel[xC][yC]`가 주어지고, 최소 바이패스 코딩된 값으로서 값 `minLevel`이 주어지면, 계수의 플래그 중 어느 것도 컨텍스트 코딩되지 않은 경우 그 계수에 대한 `minLevel`은 0이다. 플래그의 모두가 컨텍스트 코딩된 경우, `minLevel`은 10과 동일하다. 변수 `locSumAbs`는 다음 의사코드에 의해 지정된 대로 도출될 수 있다:

[0102] `locSumAbs = 0`

[0103] `if (xC > 0)`

[0104] `locSumAbs += Abs(TransCoeffLevel[ xC - 1][ yC ])`

[0105] `if (yC > 0)`

[0106] `locSumAbs += Abs(TransCoeffLevel[ xC ] [ yC -1])`

[0107] `locSumAbs = locSumAbs - 2*minLevel`

[0108] `locSumAbs = Clip3 ( 0, 31, LocSumAbs )`

[0109] 도 18은 본 개시의 일부 실시예에 따른, 최소 바이패스 코딩된 값이 0과 동일할 때 라이스 파라미터에 대한 예시적인 룩업 테이블을 예시한다. 도 18에 도시된 바와 같이, 라이스 파라미터는 변수 `cRiceParam`에 의해 나타날 수 있다. 절대치 합계의 위치는 변수 `locSumAbs`에 의해 나타날 수 있다.

[0110] 일부 실시예에서, 두 개의 별도의 룩업 테이블은 `minLevel`을 기초로 라이스 파라미터 도출을 위해 사용될 수 있다. `minLevel`이 0과 동일한 경우, 도 18에 도시된 테이블이 사용될 수 있다. `minLevel`이 0과 동일하지 않은 경우, 도 13에 도시된 테이블이 사용될 수 있다.

[0111] 일부 실시예에서, 라이스 파라미터의 도출은 임의의 룩업 테이블도 요구하지 않는다. 예를 들어, `cRiceParam`은 다음과 같이 도출될 수 있다:

[0112] `cRiceParam = (locSumAbs + offset) >> 3`

[0113] 위의 식에서, 오프셋(`offset`)은 미리 정의된 상수일 수 있으며 오프라인 트레이닝에 의해 결정될 수 있다. 오프셋 값의 일 예시는 4이다.

[0114] 일부 실시예에서, 오프셋 값은 색상 성분에 의존한다. 예를 들어, 오프셋 값은 루마에 대해 4, 및 크로마에 대해 0일 수 있다.

[0115] 일부 실시예에서, 오프셋 값은 프레임 유형에 의존한다. 예를 들어, 오프셋 값은 인트라 프레임에 대해 4, 및 인터 프레임에 대해 0일 수 있다.

[0116] 일부 실시예에서, VVC(예를 들어, VVC 7)에서, 변환 스킵 모드는 루마 및 크로마 성분 모두에 대해 허용되고, 두 타입의 성분은 동일한 컨텍스트 변수를 공유할 수 있다. 컨텍스트 변수는 최근에 디코딩된 bins를 포함하는 식에 의해 bins의 적응적 이진 산술 디코딩 프로세스에 대해 지정된 변수일 수 있다. 하지만, 루마 및 크로마 블록의 신호 통계는 상이할 수 있다. 결과적으로, 본 개시의 일부 실시예에서, 상이한 컨텍스트 변수가 루마 및 크로마 성분에 대해 사용될 수 있다. 컨텍스트 모델의 제안된 확장에 의해 영향을 받는 신호 요소는 중요도 계수 플래그(예를 들어, `sig_coeff_flag`), `abs_level_gtx_flag[n][j]`(예를 들어, `j = 0` 내지 4), 패리티 플래그(예를 들어, `par_level_flag`), 신호 계수 부호 플래그(예를 들어, `coeff_sign_flag`) 및 `coded_sub_block_flag`(예를 들어, 도 3에 도시된 `coded_sub_block_flag`)를 포함할 수 있다.

[0117] VVC(예를 들어, VVC 7)에서, 3개의 컨텍스트 변수는 변환 스킵 모드의 `sig_coeff_flag`를 코딩하기 위해 사용될 수 있다. 일부 실시예에서, 총 6개의 컨텍스트 변수(예를 들어, 루마에 대해 3개 및 크로마에 대해 3개)는 변환 스킵 모드의 `sig_coeff_flag`를 코딩하기 위해 사용될 수 있다. 변환 스킵 모드의 `sig_coeff_flag`를 코딩하기 위한 컨텍스트 인덱스는 이웃(예를 들어, 상단 및 좌측 이웃)의 중요도 계수의 수로부터 도출될 수 있다. 일부 실시예에서, 컨텍스트 인덱스는 컨텍스트 변수의 식별자를 지칭할 수 있다. 예를 들어, 6개의 컨텍스트 변수가 이용 가능한 경우, 제1 컨텍스트 변수의 컨텍스트 인덱스는 0일 수 있고, 제2 컨텍스트 변수의 컨텍스트 인덱스는 1일 수 있다. 이 프로세스에 대한 입력은 색상 성분 인덱스 `cIdx`, 루마 위치(`x0`, `y0`) 및 현재 계수 스캔 위치

(xC, yC)일 수 있다. 루마 위치(x0, y0)는 현재 픽처의 특정 샘플(예를 들어, 좌측-상단 샘플)에 대한 현재 변환 블록의 특정 샘플(예를 들어, 좌측-상단 샘플)을 지정할 수 있다. 이 프로세스의 출력은 코딩 인덱스 변수 ctxInc일 수 있다. 일부 실시예에서, 변수 ctxInc는 다음 의사코드에 따라 도출될 수 있다:

```
[0118] locNumSig = 0
[0119] if (xC > 0)
[0120]   locNumSig += sig_coeff_flag[xC - 1][yC]
[0121] if (yC > 0)
[0122]   locNumSig += sig_coeff_flag[xC][yC - 1]
[0123] if (cIdx == 0) // 예를 들어, 루마 성분 ctxInc는 다음과 같이 도출될 수 있다:
[0124]   ctxInc = locNumSig
[0125] else // 예를 들어, 크로마 성분 ctxInc는 다음과 같이 도출될 수 있다.
[0126]   ctxInc = locNumSig + 3
[0127] VVC(예를 들어, VVC 7)에서, 4개의 컨텍스트 변수는 변환 스킵 모드의 abs_level_gtx_flag[n][0]을 코딩하기
  위해 사용될 수 있다. 일부 실시예에서, 변환 스킵 모드의 abs_level_gtx_flag[n][0]을 코딩하기 위해 총 8개의
  컨텍스트 변수(루마에 대해 4개 및 크로마에 대해 4개)가 사용될 수 있다. 변환 스킵의
  abs_level_gtx_flag[n][0] 코딩을 위한 컨텍스트 인덱스는 이웃(예를 들어, 상단 및 좌측 이웃)의 중요도 계수
  의 수로부터 도출될 수 있다. 이 프로세스에 대한 입력은 색상 성분 인덱스 cIdx, 루마 위치(x0, y0) 및 현재
  계수 스캔 위치(xC, yC)일 수 있다. 루마 위치(x0, y0)는 현재 픽처의 특정 샘플(예를 들어, 좌측-상단 샘플)에
  대한 현재 변환 블록의 특정 샘플(예를 들어, 좌측-상단 샘플)을 지정할 수 있다. 이 프로세스의 출력은 코딩
  인덱스 변수 ctxInc일 수 있다. 일부 실시예에서, 변수 ctxInc는 다음 의사코드에 따라 도출될 수 있다:
[0128] if (BdpcmFlag[x0][y0][cIdx] == 1)
[0129]   ctxInc = 3
[0130] else if (xC > 0 and yC > 0)
[0131]   ctxInc = sig_coeff_flag[xC - 1][yC] + sig_coeff_flag[xC][yC - 1]
[0132] else if (xC > 0)
[0133]   ctxInc = sig_coeff_flag[xC - 1][yC]
[0134] else if (yC > 0)
[0135]   ctxInc = sig_coeff_flag[xC][yC - 1]
[0136] else
[0137]   ctxInc = 0
[0138] if (cIdx > 0)
[0139]   ctxInc = ctxInc + 4
[0140] VVC(예를 들어, VVC 7)에서, 1개의 컨텍스트 변수는 변환 스킵 모드의 par_level_flag를 코딩하기 위해 사용될
  수 있다. 일부 실시예에서, 변환 스킵 모드의 par_level_flag를 코딩하기 위해 총 2개의 컨텍스트 변수(루마에
  대해 1개 및 크로마에 대해 1개)가 사용될 수 있다. 변환 스킵의 par_level_flag를 코딩하기 위한 컨텍스트 인
  텍스는 다음과 같이 도출될 수 있다: 이 프로세스에 대한 입력은 색상 성분 인덱스 cIdx일 수 있다. 이 프로세
  스의 출력은 코딩 인덱스 변수 ctxInc이다. 일부 실시예에서, 변수 ctxInc는 다음 의사코드에 따라 도출될 수
  있다:
[0141] if (cIdx == 0)
[0142]   ctxInc = 0
```

[0143] else

[0144] ctxInc = 1

[0145] 일부 실시예에서, 루마 및 크로마에 대한 별도의 컨텍스트 변수가 변환 스킵 모드의 abs\_level\_gtx\_flag를 코딩하기 위해 사용될 수 있다. 변환 스킵의 abs\_level\_gtx\_flag를 코딩하기 위한 컨텍스트 인덱스는 다음의 방식으로 도출될 수 있다. 이 프로세스에 대한 입력은 색상 성분 인덱스 cIdx일 수 있다. 이 프로세스의 출력은 코딩 인덱스 변수 ctxInc일 수 있다. 일부 실시예에서, 변수 ctxInc는 다음 방식으로 도출될 수 있다:  $j > 0$ 인 선택 요소 abs\_level\_gtx\_flag[n][j]의 컨텍스트 인덱스는  $ctxInc = j - 1$ 로서 도출되고; 크로마 성분(예를 들어, 0보다 큰 cIdx)에 대해, 컨텍스트 인덱스는  $ctxInc = ctxInc + 4$ 로 증분된다.

[0146] VVC(예를 들어, VVC 7)에서, 6개의 컨텍스트 변수는 변환 스킵 모드의 coeff\_sign\_flag를 코딩하는데 사용될 수 있다. 일부 실시예에서, 변환 스킵 모드의 coeff\_sign\_flag를 코딩하기 위해 총 12개의 컨텍스트 변수(루마에 대해 6개 및 크로마에 대해 6개)가 사용될 수 있다. 변환 스킵 모드의 coeff\_sign\_flag를 코딩하기 위한 컨텍스트 인덱스는 이웃(예를 들어, 상단 및 좌측 이웃)의 coeff\_sign\_flag로부터 도출될 수 있다. 이 프로세스에 대한 입력은 색상 성분 인덱스 cIdx, 루마 위치( $x_0, y_0$ ) 및 현재 계수 스캔 위치( $x_C, y_C$ )일 수 있다. 루마 위치( $x_0, y_0$ )는 현재 픽처의 특정 샘플(예를 들어, 좌측-상단 샘플)에 대한 현재 변환 블록의 특정 샘플(예를 들어, 좌측-상단 샘플)을 지정할 수 있다. 이 프로세스의 출력은 코딩 인덱스 변수 ctxInc일 수 있다. 일부 실시예에서, 변수 leftSign 및 aboveSign은 다음 의사코드에 따라 도출될 수 있다:

[0147] if ( $x_C == 0$ )

[0148] leftSign = 0

[0149] else

[0150] leftSign = CoeffSignLevel[ $x_C - 1$ ][ $y_C$ ]

[0151] if ( $y_C == 0$ )

[0152] aboveSign = 0

[0153] else

[0154] aboveSign = CoeffSignLevel[ $x_C$ ][ $y_C - 1$ ]

[0155] 일부 실시예에서, 변수 ctxInc는 다음 의사코드에 따라 도출될 수 있다:

[0156] if ((leftSign == 0 and aboveSign == 0) or (leftSign == aboveSign))

[0157] if (BdpcmFlag[ $x_0$ ][ $y_0$ ][cIdx] == 0)

[0158] ctxInc = 0

[0159] else

[0160] ctxInc = 3

[0161] else if (leftSign >= 0 and aboveSign >= 0)

[0162] if (BdpcmFlag[ $x_0$ ][ $y_0$ ][cIdx])

[0163] ctxInc = 1

[0164] else

[0165] ctxInc = 4

[0166] else

[0167] if (BdpcmFlag[ $x_0$ ][ $y_0$ ][cIdx])

[0168] ctxInc = 2

[0169] else

[0170]       ctxInc = 5

[0171]       if (cIdx > 0)

[0172]       ctxInc = ctxInc + 6

[0173]       VVC(예를 들어, VVC 7)에서, 변환 스킵 모드의 coded\_sub\_block\_flag를 코딩하기 위해 3개의 컨텍스트 변수가 사용될 수 있다. coded\_sub\_block\_flag는 서브-블록에서의 변환 계수 레벨이 0과 동일한지를 지정하는 서브-블록 플래그일 수 있다. 예를 들어, coded\_sub\_block\_flag[ xS ][ yS ]가 0과 동일한 경우, 위치 (xS, yS)에서 서브블록의 변환 계수 레벨은 0과 동일한 것으로 추론된다. coded\_sub\_block\_flag[ xS ][ yS ]가 1과 동일한 경우, (xS, yS) 위치에서 서브-블록의 변환 계수 레벨 중 적어도 하나는 0이 아닌 값을 갖는다. 일부 실시예에서, 변환 스킵 모드의 coded\_sub\_block\_flag를 코딩하기 위해 총 6개의 컨텍스트 변수(루마에 대해 3개 및 크로마에 대해 3개)가 사용될 수 있다. 변환 스킵 모드의 coded\_sub\_block\_flag를 코딩하기 위한 컨텍스트 인덱스는 상단 및 좌측 이웃의 coded\_sub\_block\_flag로부터 도출될 수 있다. 이 프로세스에 대한 입력은 색상 성분 인덱스 cIdx, 루마 위치(x0, y0), 현재 서브-블록 스캔 위치(xS, yS), 선택 요소 coded\_sub\_block\_flag의 이전에 디코딩된 빈, 변환 블록 폭 log2TbWidth 및 변환 블록 높이 log2TbHeight의 이전 로그일 수 있다. 루마 위치 (x0, y0)는 현재 픽처의 좌측-상단 샘플에 상대적인 현재 변환 블록의 좌측-상단 샘플을 지정할 수 있다. 이 프로세스의 출력은 코딩 인덱스 변수 ctxInc일 수 있다. 일부 실시예에서, 변수 log2SbWidth 및 log2SbHeight는 다음 의사코드에 따라 도출될 수 있다:

[0174]       if (minimum (log2TbWidth, log2TbHeight) < 2)

[0175]       log2SbWidth = 1

[0176]       else

[0177]       log2SbHeight = log2SbWidth

[0178]       일부 실시예에서, 변수 log2SbWidth 및 log2SbHeight는 다음 의사코드에 따라 수정될 수 있다:

[0179]       if (log2TbWidth < 2 and cIdx == 0)

[0180]       log2SbWidth = log2TbWidth

[0181]       log2SbHeight = 4 - log2SbWidth

[0182]       else if (log2TbHeight < 2 and cIdx == 0)

[0183]       log2SbHeight = log2TbHeight

[0184]       log2SbWidth = 4 - log2SbHeight

[0185]       일부 실시예에서, 변수 csbfCtx는 0으로 초기화되고 다음 의사코드에 따라 수정될 수 있다:

[0186]       if (xS > 0)

[0187]       csbfCtx = csbfCtx + coded\_sub\_block\_flag[xS - 1][yS]

[0188]       if (yS > 0)

[0189]       csbfCtx = csbfCtx + coded\_sub\_block\_flag[xS][yS - 1]

[0190]       일부 실시예에서, 컨텍스트 인덱스 변수 ctxInc는 다음 의사코드에 따라 색상 성분 인덱스 cIdx 및 csbfCtx를 사용하여 도출될 수 있다:

[0191]       if (cIdx == 0)

[0192]       ctxInc = csbfCtx

[0193]       else

[0194]       ctxInc = 3 + csbfCtx

[0195]       일부 실시예에서, 명령어를 포함하는 비밀시적 컴퓨터-판독가능 저장 매체가 또한 제공되고, 명령어는 상술한 방법을 수행하기 위한 (개시된 인코더 및 디코더와 같은) 디바이스에 의해 실행될 수 있다. 비밀시적 매체의 일

반적인 형태는 예를 들어, 플로피 디스크, 플렉서블 디스크(flexible disk), 하드 디스크, 솔리드 스테이트 드라이브(solid state drive), 자기 테이프 또는 임의의 다른 자기 데이터 저장 매체, CD-ROM, 임의의 다른 광학 데이터 저장 매체, 구멍의 패턴을 갖는 임의의 물리적 매체, RAM, PROM 및 EPROM, FLASH-EPROM 또는 임의의 다른 플래시 메모리, NVRAM, 캐시, 레지스터, 임의의 다른 메모리 칩 또는 카트리지 및 이들의 네트워크화된 버전을 포함한다. 디바이스는 하나 이상의 프로세서(CPU), 입/출력 인터페이스, 네트워크 인터페이스 및/또는 메모리를 포함할 수 있다.

[0196] 본원에서 "제1" 및 "제2"와 같은 관계 용어는 엔티티 또는 동작을 다른 엔티티 또는 동작과 구분하기 위해서만 사용되며, 이러한 엔티티 또는 동작 간의 임의의 실제 관계 또는 시퀀스를 요구하거나 암시하지 않는다. 더욱이, 포함하는(comprising), "갖는(having)", "포함하는(containing)" 및 "포함하는(including)"이라는 단어 및 다른 유사한 형태는 그 의미가 동등하고, 이러한 단어 중 어느 하나를 따르는 항목 또는 항목들이 이러한 항목 또는 항목들의 철저한 열거(exhaustive listing)를 의미하지 않거나 또는 열거된 항목 또는 항목들만으로 제한되는 것을 의미하지 않는다는 점에서, 제약을 두지 않는 것으로(open-ended) 의도된다.

[0197] 본원에서 사용된, 달리 구체적으로 언급되지 않는 한, "또는"이란 용어는 실행 불가능한 경우를 제외하고 모든 가능한 조합을 포함한다. 예를 들어, 데이터베이스가 A 또는 B가 포함될 수 있다고 명시되어 있으면, 달리 구체적으로 명시되지 않거나 실행 불가능한 경우를 제외하고, 데이터베이스는 A 또는 B, 또는 A 및 B를 포함할 수 있다. 제2 예시로서, 데이터베이스가 A, B 또는 C를 포함할 수 있으면, 달리 구체적으로 명시되지 않거나 실행 불가능하지 않는 한, 데이터베이스는 A 또는 B 또는 C, 또는 A 및 B, 또는 A 및 C, 또는 B 및 C, 또는 A 및 B 및 C를 포함할 수 있다.

[0198] 상술한 실시예는 하드웨어 또는 소프트웨어(프로그램 코드), 또는 하드웨어 및 소프트웨어의 조합에 의해 구현될 수 있다는 것이 인식된다. 소프트웨어로 구현되는 경우, 이는 상술한 컴퓨터 판독 가능 매체에 저장될 수 있다. 소프트웨어는 프로세서에 의해 실행될 때 개시된 방법을 수행할 수 있다. 본 개시에서 설명된 컴퓨팅 유닛 및 다른 기능 유닛은 하드웨어 또는 소프트웨어, 또는 하드웨어 및 소프트웨어의 조합에 의해 구현될 수 있다. 통상의 기술자는 또한 위에서 설명된 모듈/유닛 중 다수가 하나의 모듈/유닛으로 결합될 수 있고, 위에서 설명된 모듈/유닛 각각이 복수의 서브-모듈/서브-유닛으로 더 분할될 수 있음을 이해할 것이다.

[0199] 전술한 명세서에서, 실시예는 구현마다 다를 수 있는 다수의 특정 세부사항을 참조로 설명되었다. 설명된 실시예의 특정 적응 및 수정이 이루어질 수 있다. 다른 실시예는 본원에 개시된 본 발명의 명세서 및 실시의 고려로부터 통상의 기술자에게 명백할 수 있다. 명세서 및 예시는 단지 예시적인 것으로 간주되어야 하는 것으로 의도되며, 본 발명의 진정한 범주 및 사상은 다음 청구범위에 의해 나타난다. 또한, 도면에 도시된 단계의 시퀀스는 단지 예시의 목적이며, 단계의 임의의 특정 시퀀스로 제한되도록 의도되지 않는다. 이와 같이, 통상의 기술자는 이러한 단계가 동일한 방법을 구현하면서 상이한 순서로 수행될 수 있음을 인식할 수 있다.

[0200] 실시예는 다음 조항을 사용하여 더 설명될 수 있다.

[0201] 1. 비디오 데이터의 인코더에 의해 구현되는 코딩 방법으로서, 상기 방법은:

[0202] 비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하는 단계를 포함하고, 스캐닝의 제1 패스는:

[0203] 변환 계수에 대한 패리티 레벨 플래그를 바이패스 코딩하는 단계 - 패리티 레벨 플래그는 변환 계수의 레벨의 절대 값의 패리티를 나타냄 -를 포함하는, 코딩 방법.

[0204] 2. 조항 1에 있어서,

[0205] 스캐닝의 제1 패스 이전에, 변환 계수의 스캐닝의 제2 패스를 수행하는 단계를 더 포함하고, 스캐닝의 제2 패스는:

[0206] 1보다 큰 플래그를 코딩하는 단계 - 1보다 큰 플래그는 절대 값이 1보다 큰지 여부를 나타냄 - 를 포함하고;

[0207] 스캐닝의 제1 패스는:

[0208] 1보다 큰 플래그가 절대 값이 1보다 크다는 것을 나타내는 것에 응답하여 패리티 레벨 플래그를 바이패스 코딩하는 단계를 더 포함하는, 코딩 방법.

[0209] 3. 조항 2에 있어서, 서브-블록은 복수의 변환 계수를 갖고, 스캐닝의 제2 패스를 수행하는 단계는:

- [0210] 컨텍스트 코딩된 빈의 수가 최대 제한에 도달할 때까지 복수의 변환 계수를 스캐닝하는 단계; 및
- [0211] 컨텍스트 코딩된 빈의 수가 최대 제한에 도달하는 것에 응답하여, 제2 패스에서 스캔되지 않은 변환 계수의 레벨의 절대 값을 바이패스 코딩하는 단계 - 바이패스 코딩하는 단계는 곱셈-라이스 코딩을 사용하여 스캔되지 않은 변환 계수의 절대 값을 이진화하는 단계를 포함함 -를 더 포함하는, 코딩 방법.
- [0212] 4. 조항 2 또는 3에 있어서, 스캐닝의 제2 패스는:
- [0213] 변환 계수에 대한 중요도 플래그를 코딩하는 단계 - 중요도 플래그는 상기 변환 계수의 레벨이 0인지를 나타냄 -; 및
- [0214] 중요도 플래그가 변환 계수의 레벨이 0이 아님을 나타내는 것에 응답하여 1보다 큰 플래그를 코딩하는 단계를 더 포함하는, 코딩 방법.
- [0215] 5. 조항 2-4 중 어느 하나에 있어서, 스캐닝의 제2 패스는:
- [0216] 1보다 큰 플래그가 절대 값이 1보다 크다는 것을 나타내는 것에 응답하여 변환 계수에 대한 3보다 큰 플래그를 코딩하는 단계 - 3보다 큰 플래그는 절대 값이 3보다 큰지 여부를 나타냄 -를 더 포함하는, 코딩 방법.
- [0217] 6. 조항 5에 있어서;
- [0218] 스캐닝의 제2 패스 이후 및 스캐닝의 제1 패스 이전에, 변환 계수의 스캐닝의 제3 패스를 수행하는 단계를 더 포함하고, 스캐닝의 제3 패스는:
- [0219] 3보다 큰 플래그가 절대 값이 3보다 크다는 것을 나타내는 것에 응답하여, 변환 계수에 대한 5보다 큰 플래그를 코딩하는 단계 - 5보다 큰 플래그는 절대 값이 5보다 큰지 여부를 나타냄 -;
- [0220] 5보다 큰 플래그가 절대 값이 5보다 크다는 것을 나타내는 것에 응답하여, 변환 계수에 대한 7보다 큰 플래그를 코딩하는 단계 - 7보다 큰 플래그는 절대 값이 7보다 큰지 여부를 나타냄 -; 및
- [0221] 7보다 큰 플래그가 절대 값이 7보다 크다는 것을 나타내는 것에 응답하여, 변환 계수에 대한 9보다 큰 플래그를 코딩하는 단계 - 9보다 큰 플래그는 절대 값이 9보다 큰지 여부를 나타냄 -를 더 포함하는, 코딩 방법.
- [0222] 7. 조항 6에 있어서, 스캐닝의 제1 패스는:
- [0223] 9보다 큰 플래그가 절대 값이 9보다 크다는 것을 나타내는 것에 응답하여 변환 계수에 대한 나머지 절대 레벨 플래그를 코딩하는 단계를 더 포함하고, 나머지 절대 레벨 플래그는 변환 계수의 레벨의 나머지 절대 값을 나타내는, 코딩 방법.
- [0224] 8. 조항 2에 있어서,
- [0225] 스캐닝의 제2 패스 이후 및 스캐닝의 제1 패스 이전에, 변환 계수의 스캐닝의 제3 패스를 수행하는 단계를 더 포함하고, 스캐닝의 제3 패스는:
- [0226] 1보다 큰 플래그가 절대 값이 1보다 크다는 것을 나타내는 것에 응답하여 변환 계수에 대한 3보다 큰 플래그를 코딩하는 단계 - 3보다 큰 플래그는 절대 값이 3보다 큰지 여부를 나타냄 -를 더 포함하는, 코딩 방법.
- [0227] 9. 조항 8에 있어서, 스캐닝의 제3 패스는:
- [0228] 3보다 큰 플래그가 절대 값이 3보다 크다는 것을 나타내는 것에 응답하여, 변환 계수에 대한 5보다 큰 플래그를 코딩하는 단계 - 5보다 큰 플래그는 절대 값이 5보다 큰지 여부를 나타냄 -;
- [0229] 5보다 큰 플래그가 절대 값이 5보다 크다는 것을 나타내는 것에 응답하여, 변환 계수에 대한 7보다 큰 플래그를 코딩하는 단계 - 7보다 큰 플래그는 절대 값이 7보다 큰지 여부를 나타냄 -; 및
- [0230] 7보다 큰 플래그가 절대 값이 7보다 크다는 것을 나타내는 것에 응답하여, 변환 계수에 대한 9보다 큰 플래그를 코딩하는 단계 - 9보다 큰 플래그는 절대 값이 9보다 큰지 여부를 나타냄 -를 더 포함하는, 코딩 방법.
- [0231] 10. 조항 9에 있어서, 스캐닝의 제1 패스는:
- [0232] 9보다 큰 플래그가 절대 값이 9보다 크다는 것을 나타내는 것에 응답하여 변환 계수에 대한 나머지 절대 레벨 플래그를 코딩하는 단계를 더 포함하고, 나머지 절대 레벨 플래그는 변환 계수의 레벨의 나머지 절대 값을 나타내는, 코딩 방법.

- [0233] 11. 조항 1-10 중 어느 하나에 있어서, 서브-블록은 복수의 변환 계수를 갖고, 스캐닝의 제1 패스는 복수의 변환 계수를 역순으로 스캔함으로써 수행되는, 코딩 방법.
- [0234] 12. 조항 11에 있어서,
- [0235] 스캐닝의 제1 패스를 수행하기 이전에 복수의 변환 계수를 플립하는 단계를 더 포함하는, 코딩 방법.
- [0236] 13. 조항 1-12 중 어느 하나에 있어서, 코딩 방법은 변환-스킵 잔차 코딩 방법인, 코딩 방법.
- [0237] 14. 조항 1-13 중 어느 하나에 있어서,
- [0238] 비디오 프레임을 수신하는 단계; 및
- [0239] 비디오 프레임을 복수의 서브-블록으로 분할하는 단계를 더 포함하는, 코딩 방법.
- [0240] 15. 비디오 데이터의 디코더에 의해 구현되는 디코딩 방법에 있어서, 상기 방법은:
- [0241] 비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하는 단계를 포함하고, 스캐닝의 제1 패스는:
- [0242] 변환 계수에 대한 패리티 레벨 플래그를 바이패스 디코딩하는 단계 - 패리티 레벨 플래그는 변환 계수의 레벨의 절대 값의 패리티를 나타냄 -를 포함하는, 디코딩 방법.
- [0243] 16. 조항 15에 있어서,
- [0244] 스캐닝의 제1 패스 이전에, 변환 계수의 스캐닝의 제2 패스를 수행하는 단계를 더 포함하고, 스캐닝의 제2 패스는:
- [0245] 1보다 큰 플래그를 디코딩하는 단계 - 1보다 큰 플래그는 절대 값이 1보다 큰지 여부를 나타냄 -를 더 포함하고;
- [0246] 스캐닝의 제1 패스는:
- [0247] 1보다 큰 플래그가 절대 값이 1보다 크다는 것을 나타내는 것에 응답하여, 제1 패스에서 패리티 레벨 플래그를 바이패스 디코딩하는 단계를 더 포함하는, 디코딩 방법.
- [0248] 17. 조항 16에 있어서, 스캐닝의 제2 패스는:
- [0249] 변환 계수에 대한 중요도 플래그를 디코딩하는 단계 - 중요도 플래그는 변환 계수의 레벨이 0인지를 나타냄 -; 및
- [0250] 중요도 플래그가 변환 계수의 레벨이 0이 아님을 나타내는 것에 응답하여 1보다 큰 플래그를 디코딩하는 단계를 더 포함하는, 디코딩 방법.
- [0251] 18. 조항 16에 있어서, 스캐닝의 제1 패스는:
- [0252] 1보다 큰 플래그가 절대 값이 1보다 크다는 것을 나타내는 것에 응답하여 변환 계수에 대한 3보다 큰 플래그를 디코딩하는 단계 - 3보다 큰 플래그는 절대 값이 3보다 큰지 여부를 나타냄 -;
- [0253] 3보다 큰 플래그가 절대 값이 3보다 크다는 것을 나타내는 것에 응답하여 변환 계수에 대한 5보다 큰 플래그를 디코딩하는 단계 - 5보다 큰 플래그는 절대 값이 5보다 큰지 여부를 나타냄 -;
- [0254] 5보다 큰 플래그가 절대 값이 5보다 크다는 것을 나타내는 것에 응답하여 변환 계수에 대한 7보다 큰 플래그를 디코딩하는 단계 - 7보다 큰 플래그는 절대 값이 7보다 큰지 여부를 나타냄 -; 및
- [0255] 7보다 큰 플래그가 절대 값이 7보다 크다는 것을 나타내는 것에 응답하여 변환 계수에 대한 9보다 큰 플래그를 디코딩하는 단계 - 9보다 큰 플래그는 절대 값이 9보다 큰지 여부를 나타냄 -를 더 포함하는, 디코딩 방법.
- [0256] 19. 조항 18에 있어서,
- [0257] 변환 계수의 스캐닝의 제3 패스를 수행하는 단계를 더 포함하고, 스캐닝의 제3 패스는:
- [0258] 9보다 큰 플래그가 절대 값이 9보다 크다는 것을 나타내는 것에 응답하여 변환 계수에 대한 나머지 절대 레벨 플래그를 디코딩하는 단계를 포함하고, 나머지 절대 레벨 플래그는 변환 계수의 레벨의 나머지 절대 값을 나타내는, 디코딩 방법.

- [0259] 20. 조항 15에 있어서, 서브-블록은 복수의 변환 계수를 갖고, 스캐닝의 제1 패스는 서브-블록의 우측-하단 코너로부터 서브-블록의 좌측-상단 코너까지 복수의 변환 계수를 스캔함으로써 수행되는, 디코딩 방법.
- [0260] 21. 조항 20에 있어서,
- [0261] 스캐닝의 제1 패스를 수행하기 이전에 복수의 변환 계수를 플립하는 단계를 더 포함하는, 디코딩 방법.
- [0262] 22. 조항 15-21 중 어느 하나에 있어서, 디코딩 방법은 변환-스킵 잔차 디코딩 방법인, 디코딩 방법.
- [0263] 23. 비디오 데이터를 인코딩하기 위한 시스템으로서, 시스템은:
- [0264] 명령어의 세트를 저장하는 메모리; 및
- [0265] 프로세서를 포함하고, 상기 프로세서는 명령어의 세트를 실행하여, 시스템이:
- [0266] 비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하는 것을 하게 하도록 구성되고, 스캐닝의 제1 패스는:
- [0267] 변환 계수에 대한 패리티 레벨 플래그를 바이패스 코딩하는 것 - 패리티 레벨 플래그는 변환 계수의 레벨의 절대 값의 패리티를 나타냄 -을 포함하는, 시스템.
- [0268] 24. 비디오 데이터를 디코딩하기 위한 시스템으로서, 상기 시스템은
- [0269] 명령어의 세트를 저장하는 메모리; 및
- [0270] 프로세서를 포함하고, 상기 프로세서는 명령어의 세트를 실행하여, 시스템이:
- [0271] 비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하는 것을 하게 하도록 구성되고, 스캐닝의 제1 패스는:
- [0272] 변환을 위한 패리티 레벨 플래그를 디코딩하는 것 - 패리티 레벨 플래그는 변환 계수의 레벨의 절대 값의 패리티를 나타냄 -을 포함하는, 시스템.
- [0273] 25. 비디오 데이터의 인코더에 의해 구현되는 코딩 방법으로서, 상기 방법은:
- [0274] 비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하는 단계 -
- [0275] 컨텍스트 코딩된 빈의 수가 최대 제한에 도달할 때 스캐닝의 제1 패스를 수행하는 단계가 중지되고,
- [0276] 서브-블록의 변환 계수의 제1 세트는 제1 패스에서 스캔되고, 및
- [0277] 스캐닝의 제1 패스는 변환 계수의 제1 세트의 각각에 대해, 변환 계수의 레벨이 0인지를 나타내는 중요도 플래그를 코딩하는 단계를 포함함 -; 및
- [0278] 서브-블록의 변환 계수의 스캐닝의 제2 패스를 수행하는 단계 - 스캐닝의 제2 패스는:
- [0279] 변환 계수의 제2 세트의 각각에 대한 레벨의 절대 값을 이진화하는 단계를 포함하고, 변환 계수의 제2 세트는 제1 패스에서 스캔되지 않음 -를 포함하는, 코딩 방법.
- [0280] 26. 조항 25에 있어서, 스캐닝의 제1 패스는:
- [0281] 제1 변환 계수의 중요도 플래그가 제1 변환 계수의 레벨이 0이 아님을 나타내는 것에 응답하여 제1 변환 계수에 대한 1보다 큰 플래그를 코딩하는 단계 - 제1 변환 계수는 변환 계수의 제1 세트 중 하나이고, 1보다 큰 플래그는 제1 변환 계수의 레벨의 절대 값이 1보다 큰지 여부를 나타냄 -를 더 포함하는, 코딩 방법.
- [0282] 27. 조항 26에 있어서, 스캐닝의 제1 패스는:
- [0283] 1보다 큰 플래그가 제1 변환 계수의 레벨의 절대 값이 1보다 크다는 것을 나타내는 것에 응답하여 제1 변환 계수에 대한 3보다 큰 플래그를 코딩하는 단계 - 3보다 큰 플래그는 제1 변환 계수의 레벨의 절대 값이 3보다 큰지 여부를 나타내고, 스캐닝의 제1 패스를 수행하는 단계는 나머지 컨텍스트 코딩된 빈의 수가 그룹 제한보다 적을 때 중지됨 -;
- [0284] 3보다 큰 플래그가 제1 변환 계수의 레벨의 절대 값이 3보다 크다는 것을 나타내는 것에 응답하여 제1 변환 계수에 대한 5보다 큰 플래그를 코딩하는 단계 - 5보다 큰 플래그는 제1 변환 계수의 레벨의 절대 값이 5보다 큰지 여부를 나타냄 -;

- [0285] 5보다 큰 플래그가 제1 변환 계수의 레벨의 절대 값이 5보다 크다는 것을 나타내는 것에 응답하여 제1 변환 계수에 대한 7보다 큰 플래그를 코딩하는 단계 - 7보다 큰 플래그는 제1 변환 계수의 레벨의 절대 값이 7보다 큰지 여부를 나타냄 -; 및
- [0286] 7보다 큰 플래그가 제1 변환 계수의 레벨의 절대 값이 7보다 크다는 것을 나타내는 것에 응답하여 제1 변환 계수에 대한 9보다 큰 플래그를 코딩하는 단계 - 9보다 큰 플래그는 제1 변환 계수 레벨의 절대 값이 9보다 큰지 여부를 나타냄 -를 더 포함하는, 코딩 방법.
- [0287] 28. 조항 27에 있어서, 스캐닝의 제2 패스는:
- [0288] 9보다 큰 플래그가 상기 제1 변환 계수의 레벨의 절대 값이 9보다 크다는 것을 나타내는 것에 응답하여 제1 변환 계수에 대한 나머지 절대 레벨 플래그를 코딩하는 단계 - 나머지 절대 레벨 플래그는 제1 변환 계수의 레벨의 나머지 절대 값을 나타냄 -를 더 포함하는, 코딩 방법.
- [0289] 29. 조항 25-28 중 어느 하나에 있어서, 제1 패스 및 제2 패스의 각각은 서브-블록의 변환 계수를 역순으로 스캔함으로써 수행되는, 코딩 방법.
- [0290] 30. 조항 29에 있어서, 방법은:
- [0291] 제1 패스를 수행하기 전에 서브-블록의 변환 계수를 플립하는 단계를 더 포함하는, 코딩 방법.
- [0292] 31. 조항 25-30 중 어느 하나에 있어서, 코딩 방법은 변환-스킵 잔차 코딩 방법인, 코딩 방법.
- [0293] 32. 조항 25-31 중 어느 하나에 있어서, 변환 계수의 제2 세트의 각각에 대한 레벨의 절대 값을 이진화하는 단계는:
- [0294] 곱셈-라이트 코드를 사용하여 절대 값을 이진화하는 단계를 더 포함하는, 코딩 방법.
- [0295] 33. 비디오 데이터의 디코더에 의해 구현되는 디코딩 방법에 있어서, 상기 방법은:
- [0296] 비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하는 단계 -
- [0297] 컨텍스트 코딩된 빈의 수가 최대 제한에 도달할 때, 스캐닝의 제1 패스를 수행하는 단계가 중지되고,
- [0298] 서브-블록의 변환 계수의 제1 세트는 제1 패스에서 스캔되고, 및
- [0299] 스캐닝의 제1 패스는 변환 계수의 제1 세트의 각각에 대해, 변환 계수의 레벨이 0인지를 나타내는 중요도 플래그를 디코딩하는 단계를 포함함 -; 및
- [0300] 서브-블록의 변환 계수의 스캐닝의 제2 패스를 수행하는 단계 - 스캐닝의 제2 패스는:
- [0301] 변환 계수의 제2 세트의 각각에 대한 레벨의 이진화된 절대 값을 디코딩하는 단계 - 변환 계수의 제2 세트는 제1 패스에서 스캔되지 않음 -를 포함하는, 디코딩 방법.
- [0302] 34. 조항 33에 있어서, 스캐닝의 제1 패스는:
- [0303] 제1 변환 계수의 중요도 플래그가 제1 변환 계수의 레벨이 0이 아님을 나타내는 것에 응답하여 제1 변환 계수에 대한 1보다 큰 플래그를 디코딩하는 단계 - 제1 변환 계수는 변환 계수의 제1 세트 중 하나이고, 1보다 큰 플래그는 제1 변환 계수의 레벨의 절대 값이 1보다 큰지 여부를 나타냄 -를 더 포함하는, 디코딩 방법.
- [0304] 35. 조항 34에 있어서, 스캐닝의 제1 패스는:
- [0305] 1보다 큰 플래그가 제1 변환 계수의 레벨의 절대 값이 1보다 크다는 것을 나타내는 것에 응답하여 제1 변환 계수에 대한 3보다 큰 플래그를 디코딩하는 단계 - 3보다 큰 플래그는 스캔되는 제1 변환 계수의 레벨의 절대 값이 3보다 큰지 여부를 나타내고, 나머지 컨텍스트 코딩된 빈의 수가 그룹 제한보다 적을 때 스캐닝의 제1 패스를 수행하는 단계가 중지됨 -;
- [0306] 3보다 큰 플래그가 제1 변환 계수의 레벨의 절대 값이 3보다 크다는 것을 나타내는 것에 응답하여 제1 변환 계수의 각각에 대한 5보다 큰 플래그를 디코딩하는 단계 - 5보다 큰 플래그는 제1 변환 계수의 레벨의 절대 값이 5보다 큰지 여부를 나타냄 -;
- [0307] 5보다 큰 플래그가 제1 변환 계수의 레벨의 절대 값이 5보다 크다는 것을 나타내는 것에 응답하여 제1 변환 계수의 각각에 대해 7보다 큰 플래그를 디코딩하는 단계 - 7보다 큰 플래그는 제1 변환 계수의 레벨의 절대 값이

7보다 큰지 여부를 나타냄 -; 및

- [0308] 7보다 큰 플래그가 제1 변환 계수의 레벨의 절대 값이 7보다 크다는 것을 나타내는 것에 응답하여 제1 변환 계수의 각각에 대한 9보다 큰 플래그를 디코딩하는 단계 - 9보다 큰 플래그는 제1 변환 계수의 레벨의 절대 값이 9보다 큰지 여부를 나타냄 -를 더 포함하는, 디코딩 방법.
- [0309] 36. 조항 35에 있어서, 스캐닝의 제2 패스는:
- [0310] 9보다 큰 플래그가 스캔되는 제1 변환 계수의 레벨의 절대 값이 9보다 크다는 것을 나타내는 것에 응답하여 제1 변환 계수에 대한 나머지 절대 레벨 플래그를 디코딩하는 단계 - 나머지 절대 레벨 플래그는 제1 변환 계수의 레벨의 나머지 절대 값을 나타냄 -를 더 포함하는, 디코딩 방법.
- [0311] 37. 조항 33-36 중 어느 하나에 있어서, 코딩 방법은 변환-스킵 잔차 코딩 방법인, 디코딩 방법.
- [0312] 38. 조항 9-13 중 어느 하나에 있어서, 변환 계수의 제2 세트의 각각에 대한 레벨의 이진화된 절대 값을 디코딩하는 단계는:
- [0313] 골롭-라이스 코드를 사용하여 절대 값을 디코딩하는 단계를 더 포함하는, 디코딩 방법.
- [0314] 39. 비디오 데이터를 인코딩하기 위한 시스템으로서,
- [0315] 명령어의 세트를 저장하는 메모리; 및
- [0316] 프로세서를 포함하고, 상기 프로세서는 명령어의 세트를 실행하여, 시스템이:
- [0317] 비디오 프레임의 서브-블록의 스캐닝 변환 계수의 제1 패스를 수행하게 하고 -
- [0318] 컨텍스트 코딩된 bin의 수가 최대 제한에 도달할 때 스캐닝의 제1 패스의 수행이 중지되고,
- [0319] 서브-블록의 변환 계수의 제1 세트는 제1 패스에서 스캔되고, 및
- [0320] 스캐닝의 제1 패스는 변환 계수의 제1 세트의 각각에 대해, 변환 계수의 레벨이 0인지를 나타내는 중요도 플래그를 코딩하는 단계를 포함함 -; 및
- [0321] 서브-블록의 변환 계수의 스캐닝의 제2 패스를 수행하게 하도록 - 스캐닝의 제2 패스는:
- [0322] 변환 계수의 제2 세트의 각각에 대한 레벨의 절대 값을 이진화하는 것을 포함하고, 변환 계수의 제2 세트는 제1 패스에서 스캔되지 않음 - 구성되는, 시스템.
- [0323] 40. 비디오 데이터를 디코딩하기 위한 시스템으로서,
- [0324] 명령어의 세트를 저장하는 메모리; 및
- [0325] 프로세서를 포함하고, 상기 프로세서는 명령어의 세트를 실행하여, 시스템이:
- [0326] 비디오 프레임의 서브-블록의 스캐닝 변환 계수의 제1 패스를 수행하게 하고 -
- [0327] 컨텍스트 코딩된 bin의 수가 최대 제한에 도달할 때, 스캐닝의 제1 패스의 수행이 중지되고,
- [0328] 서브-블록의 변환 계수의 제1 세트는 제1 패스에서 스캔되고, 및
- [0329] 스캐닝의 제1 패스는 변환 계수의 제1 세트의 각각에 대해, 변환 계수의 레벨이 0인지를 나타내는 중요도 플래그를 디코딩하는 것을 포함함 -; 및
- [0330] 서브-블록의 변환 계수의 스캐닝의 제2 패스를 수행하게 하도록 구성되고, 스캐닝의 제2 패스는:
- [0331] 변환 계수의 제2 세트의 각각에 대한 레벨의 이진화된 절대 값을 디코딩하는 것 - 변환 계수의 제2 세트는 제1 패스에서 스캔되지 않음 -을 포함하는, 시스템.
- [0332] 41. 비디오 데이터의 인코더에 의해 구현되는 코딩 방법으로서, 상기 방법은:
- [0333] 비디오 프레임의 루마 성분에 대한 컨텍스트 변수의 제1 세트를 생성하는 단계;
- [0334] 비디오 프레임의 크로마 성분에 대한 컨텍스트 변수의 제2 세트를 생성하는 단계;
- [0335] 비디오 프레임의 서브-블록을 생성하는 단계; 및
- [0336] 컨텍스트 변수의 제1 세트 및 컨텍스트 변수의 제2 세트에 따라 서브-블록의 변환 계수의 제1 세트를 코딩하는

단계를 포함하는, 코딩 방법.

- [0337] 42. 조항 41에 있어서, 변환 계수의 제1 세트를 코딩하는 단계는:
- [0338] 컨텍스트 변수의 제1 세트로부터의 3개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 3개의 컨텍스트 변수에 따라 변환 계수의 제1 세트의 변환 계수에 대한 중요도 플래그를 코딩하는 단계 - 중요도 플래그는 변환 계수의 레벨이 0인지를 나타냄 -를 포함하는, 코딩 방법.
- [0339] 43. 조항 42에 있어서,
- [0340] 컨텍스트 변수의 제1 세트로부터의 3개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 3개의 컨텍스트 변수를 생성하는 단계 - 생성하는 단계는 색상 성분 인덱스, 비디오 프레임에 관련된 서브-블록의 위치를 지정하는 루마 위치, 및 현재 계수 스캔 위치를 기초로 함 -를 더 포함하는, 코딩 방법.
- [0341] 44. 조항 41-43 중 어느 하나에 있어서, 변환 계수의 제1 세트를 코딩하는 단계는:
- [0342] 컨텍스트 변수의 제1 세트로부터의 4개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 4개의 컨텍스트 변수에 따라 변환 계수의 제1 세트의 변환 계수에 대한  $x$ 보다 큰 플래그를 코딩하는 단계 -  $x$ 보다 큰 플래그는 변환 계수의 레벨의 절대 값이 숫자  $x$ 보다 큰지 여부를 나타냄 -를 포함하는, 코딩 방법.
- [0343] 45. 조항 44에 있어서,
- [0344] 컨텍스트 변수의 제1 세트로부터의 4개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 4개의 컨텍스트 변수를 생성하는 단계 - 생성하는 단계는 색상 성분 인덱스, 비디오 프레임에 관련된 서브-블록의 위치를 지정하는 루마 위치 및 현재 계수 스캔 위치를 기초로 함 -를 더 포함하는, 코딩 방법.
- [0345] 46. 조항 41-45 중 어느 하나에 있어서, 변환 계수의 제1 세트를 코딩하는 단계는:
- [0346] 컨텍스트 변수의 제1 세트로부터의 1개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 1개의 컨텍스트 변수에 따라 변환 계수의 제1 세트의 변환 계수에 대한 패리티 플래그를 코딩하는 단계 - 패리티 플래그는 변환 계수의 레벨의 절대 값의 패리티를 나타냄 -를 포함하는, 코딩 방법.
- [0347] 47. 조항 46에 있어서,
- [0348] 색상 성분 인덱스에 따라, 컨텍스트 변수의 제1 세트로부터의 1개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 1개의 컨텍스트 변수를 생성하는 단계를 더 포함하는, 코딩 방법.
- [0349] 48. 조항 41-47 중 어느 하나에 있어서, 변환 계수의 제1 세트를 코딩하는 단계는:
- [0350] 컨텍스트 변수의 제1 세트로부터의 6개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 6개의 컨텍스트 변수에 따라 변환 계수의 제1 세트의 변환 계수에 대한 계수 부호 플래그를 코딩하는 단계 - 계수 부호 플래그는 변환 계수의 값의 부호를 나타냄 -를 포함하는, 코딩 방법.
- [0351] 49. 조항 48에 있어서,
- [0352] 컨텍스트 변수의 제1 세트로부터의 6개의 컨텍스트 변수, 및 컨텍스트 변수의 제2 세트로부터의 6개의 컨텍스트 변수를 생성하는 단계 - 생성하는 단계는 비디오 프레임의 이웃의 중요도 계수의 수, 비디오 프레임에 관련된 서브-블록의 위치를 지정하는 루마 위치 및 현재 계수 스캔 위치 -를 더 포함하는, 코딩 방법.
- [0353] 50. 조항 41-49 중 어느 하나에 있어서, 변환 계수의 제1 세트를 코딩하는 단계는:
- [0354] 컨텍스트 변수의 제1 세트로부터의 3개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 3개의 컨텍스트 변수에 따라 변환 계수의 제1 세트의 변환 계수에 대한 서브-블록 플래그를 코딩하는 단계를 포함하는, 코딩 방법.
- [0355] 51. 조항 50에 있어서,
- [0356] 컨텍스트 변수의 제1 세트로부터의 3개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 3개의 컨텍스트 변수를 생성하는 단계 - 생성하는 단계는 비디오 프레임의 이웃의 중요도 계수의 수, 비디오 프레임에 대한 서브-블록의 위치를 지정하는 루마 위치 및 현재 계수 스캔 위치를 기초로 함 -를 더 포함하는, 코딩 방법.
- [0357] 52. 비디오 처리 방법으로서,

- [0358] 비디오 비트-스트림을 수신하는 단계;
- [0359] 비디오 비트-스트림을 복수의 서브-블록으로 분할하는 단계;
- [0360] 서브-블록의 루마 성분에 대한 컨텍스트 변수의 제1 세트를 생성하는 단계;
- [0361] 서브-블록의 크로마 성분에 대한 컨텍스트 변수의 제2 세트를 생성하는 단계; 및
- [0362] 컨텍스트 변수의 제1 세트 및 컨텍스트 변수의 제2 세트에 따라 서브-블록의 변환 계수의 제1 세트를 컨텍스트 코딩하는 단계를 포함하는, 비디오 처리 방법.
- [0363] 53. 비디오 데이터의 디코딩에 의해 구현되는 디코딩 방법에 있어서, 상기 방법은:
- [0364] 비디오 프레임을 수신하는 단계;
- [0365] 비디오 프레임을 복수의 서브-블록으로 분할하는 단계;
- [0366] 비디오 프레임의 루마 성분에 대한 컨텍스트 변수의 제1 세트를 생성하는 단계;
- [0367] 비디오 프레임의 크로마 성분에 대한 컨텍스트 변수의 제2 세트를 생성하는 단계; 및
- [0368] 컨텍스트 변수의 제1 세트 및 컨텍스트 변수의 제2 세트에 따라 서브-블록의 변환 계수의 제1 세트를 디코딩하는 단계를 포함하는, 디코딩 방법.
- [0369] 54. 조항 53에 있어서, 변환 계수들의 제1 세트를 디코딩하는 단계는:
- [0370] 컨텍스트 변수의 제1 세트로부터의 3개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 3개의 컨텍스트 변수에 따라 변환 계수의 제1 세트의 변환 계수에 대한 중요도 플래그를 디코딩하는 단계 - 중요도 플래그는 변환 계수의 레벨이 0인지를 나타냄 -를 포함하는, 디코딩 방법.
- [0371] 55. 조항 54에 있어서,
- [0372] 컨텍스트 변수의 제1 세트로부터의 3개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 3개의 컨텍스트 변수를 생성하는 단계 - 생성하는 단계는 색상 성분 인덱스, 비디오 프레임의 좌측-상단 샘플에 대한 현재 변환 블록의 좌측-상단 샘플을 지정하는 루마 위치, 및 현재 계수 스캔 위치를 기초로 함 -를 더 포함하는, 디코딩 방법.
- [0373] 56. 조항 53-55 중 어느 하나에 있어서, 변환 계수의 제1 세트를 디코딩하는 단계는:
- [0374] 컨텍스트 변수의 제1 세트로부터의 4개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 4개의 컨텍스트 변수에 따라 변환 계수의 제1 세트의 변환 계수에 대한 x보다 큰 플래그를 디코딩하는 단계를 포함하는, 디코딩 방법.
- [0375] 57. 조항 56에 있어서,
- [0376] 컨텍스트 변수의 제1 세트로부터의 4개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 4개의 컨텍스트 변수를 생성하는 단계 - 생성하는 단계는 색상 성분 인덱스, 비디오 프레임의 좌측-상단 샘플에 대한 현재 변환 블록의 좌측-상단 샘플을 지정하는 루마 위치, 및 현재 계수 스캔 위치를 기초로 함 -를 더 포함하는, 디코딩 방법.
- [0377] 58. 조항 53-57 중 어느 하나에 있어서, 변환 계수의 제1 세트를 디코딩하는 단계는:
- [0378] 컨텍스트 변수의 제1 세트로부터의 1개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 1개의 컨텍스트 변수에 따라 변환 계수의 제1 세트의 변환 계수에 대한 패리티 플래그를 디코딩하는 단계를 포함하는, 디코딩 방법.
- [0379] 59. 조항 58에 있어서,
- [0380] 컨텍스트 변수의 제1 세트로부터의 1개의 컨텍스트 변수를 생성하고 컨텍스트 변수의 제2 세트로부터의 1개의 컨텍스트 변수를 생성하는 단계 - 생성하는 단계는 색상 성분 인덱스를 기초로 함 -를 더 포함하는, 디코딩 방법.
- [0381] 60. 조항 53-59 중 어느 하나에 있어서, 변환 계수의 제1 세트를 디코딩하는 단계는:
- [0382] 컨텍스트 변수의 제1 세트로부터의 6개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 6개의 컨텍스트

변수에 따라 변환 계수의 제1 세트의 변환 계수에 대한 컨텍스트 계수 부호 플래그를 디코딩하는 단계를 포함하는, 디코딩 방법.

- [0383] 61. 조항 60에 있어서,
- [0384] 컨텍스트 변수의 제1 세트로부터의 6개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 6개의 컨텍스트 변수를 생성하는 단계 - 생성하는 단계는 비디오 프레임의 상단 및 좌측 이웃의 중요도 계수의 수, 비디오 프레임의 좌측-상단 샘플에 대한 현재 변환 블록의 좌측-상단 샘플을 지정하는 루마 위치, 및 현재 계수 스캔 위치를 기초로 함 - 를 더 포함하는, 디코딩 방법.
- [0385] 62. 조항 53-61 중 어느 하나에 있어서, 변환 계수의 제1 세트를 디코딩하는 단계는:
- [0386] 컨텍스트 변수의 제1 세트로부터의 3개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 3개의 컨텍스트 변수에 따라 변환 계수의 제1 세트의 변환 계수에 대한 서브-블록 플래그를 디코딩하는 단계를 포함하는, 디코딩 방법.
- [0387] 63. 조항 62에 있어서,
- [0388] 컨텍스트 변수의 제1 세트로부터의 3개의 컨텍스트 변수 및 컨텍스트 변수의 제2 세트로부터의 3개의 컨텍스트 변수를 생성하는 단계 - 생성하는 단계는 비디오 프레임의 상단 및 좌측 이웃의 중요도 계수의 수, 비디오 프레임의 좌측-상단 샘플에 대한 현재 변환 블록의 좌측-상단 샘플을 지정하는 루마 위치 및 현재 계수 스캔 위치를 기초로 함 - 를 더 포함하는, 디코딩 방법.
- [0389] 64. 비디오 데이터를 인코딩하기 위한 시스템으로서,
- [0390] 명령어의 세트를 저장하는 메모리; 및
- [0391] 프로세서를 포함하고, 상기 프로세서는 명령어의 세트를 실행하여, 시스템이:
- [0392] 비디오 프레임의 루마 성분에 대한 컨텍스트 변수의 제1 세트를 생성하고;
- [0393] 비디오 프레임의 크로마 성분에 대한 컨텍스트 변수의 제2 세트를 생성하고;
- [0394] 비디오 프레임의 서브-블록을 생성하고; 및
- [0395] 컨텍스트 변수의 제1 세트 및 컨텍스트 변수의 제2 세트에 따라 서브-블록의 변환 계수의 제1 세트를 코딩하게 하도록 구성되는, 시스템.
- [0396] 65. 비디오 데이터를 디코딩하기 위한 시스템으로서,
- [0397] 명령어의 세트를 저장하는 메모리; 및
- [0398] 프로세서를 포함하고, 상기 프로세서는 명령어의 세트를 실행하여, 시스템이:
- [0399] 비디오 프레임을 수신하고;
- [0400] 비디오 프레임을 복수의 서브-블록으로 분할하고;
- [0401] 비디오 프레임의 루마 성분에 대한 컨텍스트 변수의 제1 세트를 생성하고;
- [0402] 비디오 프레임의 크로마 성분에 대한 컨텍스트 변수의 제2 세트를 생성하고; 및
- [0403] 컨텍스트 변수의 제1 세트 및 컨텍스트 변수의 제2 세트에 따라 서브-블록의 변환 계수의 제1 세트를 디코딩하게 하도록 구성되는, 시스템.
- [0404] 66. 장치가 비디오 데이터를 인코딩하기 위한 방법을 개시하게 하는 장치의 하나 이상의 프로세서에 의해 실행 가능한 명령어의 세트를 저장하는 비일시적 컴퓨터 판독가능 매체로서, 상기 방법은:
- [0405] 비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하는 단계를 포함하고, 스캐닝의 제1 패스는:
- [0406] 변환 계수에 대한 패리티 레벨 플래그를 바이패스 코딩하는 단계 - 패리티 레벨 플래그는 변환 계수의 레벨의 절대 값의 패리티를 나타냄 - 를 포함하는, 비일시적 컴퓨터 판독가능 매체.
- [0407] 67. 장치가 비디오 데이터를 디코딩하기 위한 방법을 개시하게 하는 장치의 하나 이상의 프로세서에 의해 실행

가능한 명령어의 세트를 저장하는 비밀시적 컴퓨터 판독가능 매체로서, 상기 방법은:

- [0408] 비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하는 단계를 포함하고, 스캐닝의 제1 패스는:
- [0409] 변환 계수에 대한 패리티 레벨 플래그를 바이패스 디코딩하는 단계 - 패리티 레벨 플래그는 변환 계수의 레벨의 절대 값의 패리티를 나타냄 -를 포함하는, 비밀시적 컴퓨터 판독가능 매체.
- [0410] 68. 장치가 비디오 데이터를 인코딩하기 위한 방법을 개시하게 하는 장치의 하나 이상의 프로세서에 의해 실행 가능한 명령어의 세트를 저장하는 비밀시적 컴퓨터 판독가능 매체로서, 상기 방법은:
- [0411] 비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하는 단계 -
- [0412] 컨텍스트 코딩된 bin의 수가 최대 제한에 도달할 때 스캐닝의 제1 패스를 수행하는 단계가 중지되고,
- [0413] 서브-블록의 변환 계수의 제1 세트는 제1 패스에서 스캔되고, 및
- [0414] 스캐닝의 제1 패스는 변환 계수의 제1 세트의 각각에 대해, 변환 계수의 레벨이 0인지를 나타내는 중요도 플래그를 코딩하는 단계를 포함함 -; 및
- [0415] 서브-블록의 변환 계수의 스캐닝의 제2 패스를 수행하는 단계 - 스캐닝의 제2 패스는:
- [0416] 변환 계수의 제2 세트의 각각에 대한 레벨의 절대 값을 이진화하는 단계를 포함하고, 변환 계수의 제2 세트는 제1 패스에서 스캔되지 않음 -를 포함하는, 비밀시적 컴퓨터 판독가능 매체.
- [0417] 69. 장치가 비디오 데이터를 디코딩하기 위한 방법을 개시하게 하는 장치의 하나 이상의 프로세서에 의해 실행 가능한 명령어의 세트를 저장하는 비밀시적 컴퓨터 판독가능 매체로서, 상기 방법은:
- [0418] 비디오 프레임의 서브-블록의 변환 계수의 스캐닝의 제1 패스를 수행하는 단계 -
- [0419] 컨텍스트 코딩된 bin의 수가 최대 제한에 도달할 때, 스캐닝의 제1 패스를 수행하는 단계가 중지되고,
- [0420] 서브-블록의 변환 계수의 제1 세트는 제1 패스에서 스캔되고, 및
- [0421] 스캐닝의 제1 패스는 변환 계수의 제1 세트의 각각에 대해, 변환 계수의 레벨이 0인지를 나타내는 중요도 플래그를 디코딩하는 단계를 포함함 -; 및
- [0422] 서브-블록의 변환 계수의 스캐닝의 제2 패스를 수행하는 단계 - 스캐닝의 제2 패스는:
- [0423] 변환 계수의 제2 세트의 각각에 대한 레벨의 이진화된 절대 값을 디코딩하는 단계 - 변환 계수의 제2 세트는 제1 패스에서 스캔되지 않음 -를 포함하는, 비밀시적 컴퓨터 판독가능 매체.
- [0424] 70. 장치가 비디오 데이터를 인코딩하기 위한 방법을 개시하게 하는 장치의 하나 이상의 프로세서에 의해 실행 가능한 명령어의 세트를 저장하는 비밀시적 컴퓨터 판독가능 매체로서, 상기 방법은:
- [0425] 비디오 프레임의 루마 성분에 대한 컨텍스트 변수의 제1 세트를 생성하는 단계;
- [0426] 비디오 프레임의 크로마 성분에 대한 컨텍스트 변수의 제2 세트를 생성하는 단계;
- [0427] 비디오 프레임의 서브-블록을 생성하는 단계; 및
- [0428] 컨텍스트 변수의 제1 세트 및 컨텍스트 변수의 제2 세트에 따라 서브-블록의 변환 계수의 제1 세트를 코딩하는 단계를 포함하는, 비밀시적 컴퓨터 판독가능 매체.
- [0429] 71. 장치가 비디오 데이터를 디코딩하기 위한 방법을 개시하게 하는 장치의 하나 이상의 프로세서에 의해 실행 가능한 명령어의 세트를 저장하는 비밀시적 컴퓨터 판독가능 매체로서, 상기 방법은:
- [0430] 비디오 프레임을 수신하는 단계;
- [0431] 비디오 프레임을 복수의 서브-블록으로 분할하는 단계;
- [0432] 비디오 프레임의 루마 성분에 대한 컨텍스트 변수의 제1 세트를 생성하는 단계;
- [0433] 비디오 프레임의 크로마 성분에 대한 컨텍스트 변수의 제2 세트를 생성하는 단계; 및
- [0434] 컨텍스트 변수의 제1 세트 및 컨텍스트 변수의 제2 세트에 따라 서브-블록의 변환 계수의 제1 세트를 디코딩하

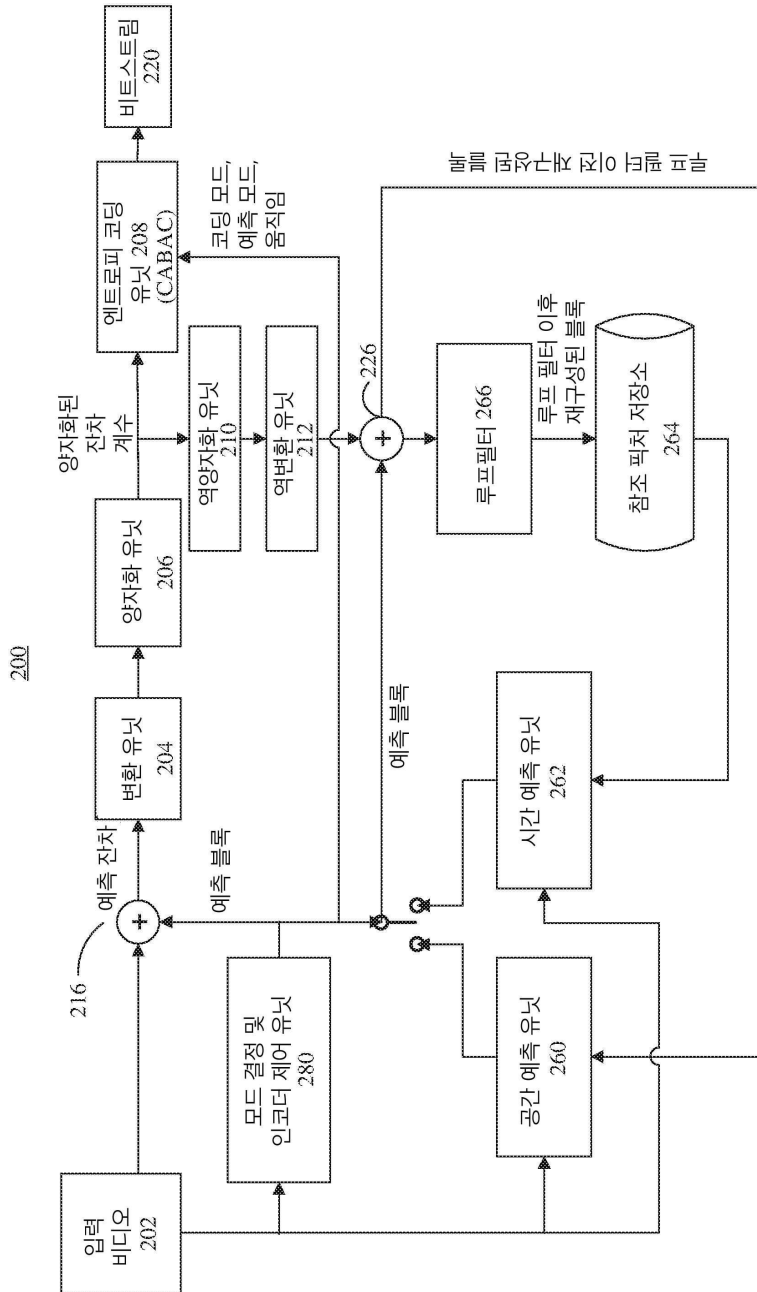
는 단계를 포함하는, 비밀시적 컴퓨터 판독가능 매체.

[0435]

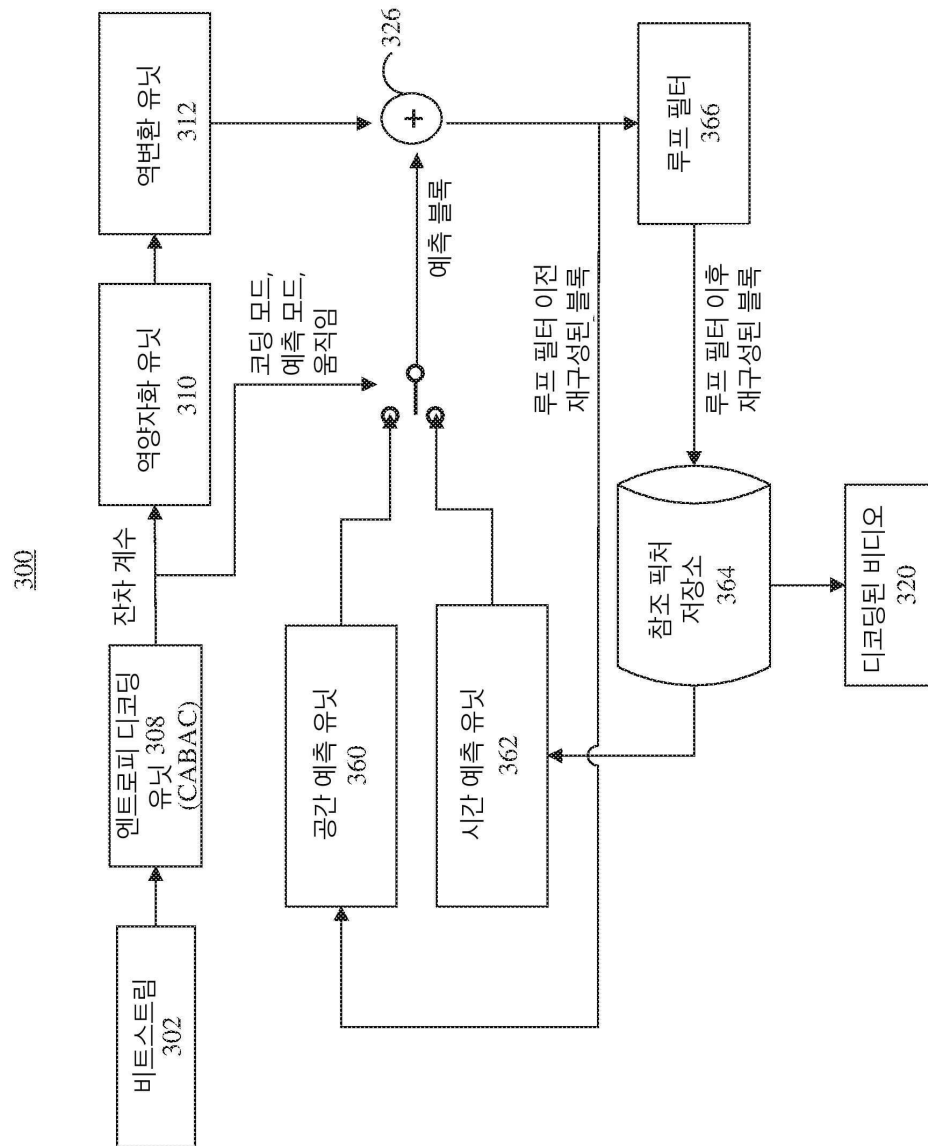
도면 및 명세서에서, 예시적인 실시예가 개시된다. 하지만, 이들 실시예에 대해 다수의 변형 및 수정이 이루어질 수 있다. 따라서, 특정한 용어가 이용되더라도, 이는 제한의 목적이 아닌 일반적이고 설명적인 의미로 사용된다.

도면

도면1



도면2



도면3a

|                                                                                                                                      |       |
|--------------------------------------------------------------------------------------------------------------------------------------|-------|
| residual_coding( x0, y0, log2TbWidth, log2TbHeight, cIdx ) {                                                                         | 설명자   |
| if( ( tu_mts_idx[ x0 ][ y0 ] > 0   <br>( cu_sbt_flag && log2TbWidth < 6 && log2TbHeight < 6 ) )<br>&& cIdx == 0 && log2TbWidth > 4 ) |       |
| log2ZoTbWidth = 4                                                                                                                    |       |
| Else                                                                                                                                 |       |
| log2ZoTbWidth = Min( log2TbWidth, 5 )                                                                                                |       |
| MaxCbs = 2 * ( 1 << log2TbWidth ) * ( 1 << log2TbHeight )                                                                            |       |
| if( tu_mts_idx[ x0 ][ y0 ] > 0   <br>( cu_sbt_flag && log2TbWidth < 6 && log2TbHeight < 6 ) )<br>&& cIdx == 0 && log2TbHeight > 4 )  |       |
| log2ZoTbHeight = 4                                                                                                                   |       |
| Else                                                                                                                                 |       |
| log2ZoTbHeight = Min( log2TbHeight, 5 )                                                                                              |       |
| if( log2TbWidth > 0 )                                                                                                                |       |
| <b>last_sig_coeff_x_prefix</b>                                                                                                       | ae(v) |
| if( log2TbHeight > 0 )                                                                                                               |       |
| <b>last_sig_coeff_y_prefix</b>                                                                                                       | ae(v) |
| if( last_sig_coeff_x_prefix > 3 )                                                                                                    |       |
| <b>last_sig_coeff_x_suffix</b>                                                                                                       | ae(v) |
| if( last_sig_coeff_y_prefix > 3 )                                                                                                    |       |
| <b>last_sig_coeff_y_suffix</b>                                                                                                       | ae(v) |
| log2TbWidth = log2ZoTbWidth                                                                                                          |       |
| log2TbHeight = log2ZoTbHeight                                                                                                        |       |
| log2SbW = ( Min( log2TbWidth, log2TbHeight ) < 2 ? 1 : 2 )                                                                           |       |
| log2SbH = log2SbW                                                                                                                    |       |
| if( log2TbWidth + log2TbHeight > 3 ) {                                                                                               |       |
| if( log2TbWidth < 2 ) {                                                                                                              |       |
| log2SbW = log2TbWidth                                                                                                                |       |
| log2SbH = 4 - log2SbW                                                                                                                |       |
| } else if( log2TbHeight < 2 ) {                                                                                                      |       |
| log2SbH = log2TbHeight                                                                                                               |       |
| log2SbW = 4 - log2SbH                                                                                                                |       |
| }                                                                                                                                    |       |
| }                                                                                                                                    |       |
| numSbCoeff = 1 << ( log2SbW + log2SbH )                                                                                              |       |
| lastScanPos = numSbCoeff                                                                                                             |       |
| lastSubBlock = ( 1 << ( log2TbWidth + log2TbHeight - ( log2SbW + log2SbH ) ) ) - 1                                                   |       |
| do {                                                                                                                                 |       |

도면3b

|                                                                                  |       |
|----------------------------------------------------------------------------------|-------|
| if( lastScanPos == 0 ) {                                                         |       |
| lastScanPos = numSbCoeff                                                         |       |
| lastSubBlock--                                                                   |       |
| }                                                                                |       |
| lastScanPos--                                                                    |       |
| xS = DiagScanOrder[ log2TbWidth - log2SbW ][ log2TbHeight - log2SbH ]            |       |
| [ lastSubBlock ][ 0 ]                                                            |       |
| yS = DiagScanOrder[ log2TbWidth - log2SbW ][ log2TbHeight - log2SbH ]            |       |
| [ lastSubBlock ][ 1 ]                                                            |       |
| xC = ( xS << log2SbW ) + DiagScanOrder[ log2SbW ][ log2SbH ][ lastScanPos ][ 0 ] |       |
| yC = ( yS << log2SbH ) + DiagScanOrder[ log2SbW ][ log2SbH ][ lastScanPos ][ 1 ] |       |
| } while( ( xC != LastSignificantCoeffX )    ( yC != LastSignificantCoeffY ) )    |       |
| QState = 0                                                                       |       |
| for( i = lastSubBlock; i >= 0; i-- ) {                                           |       |
| startQStateSb = QState                                                           |       |
| xS = DiagScanOrder[ log2TbWidth - log2SbW ][ log2TbHeight - log2SbH ]            |       |
| [ i ][ 0 ]                                                                       |       |
| yS = DiagScanOrder[ log2TbWidth - log2SbW ][ log2TbHeight - log2SbH ]            |       |
| [ i ][ 1 ]                                                                       |       |
| inferSbDcSigCoeffFlag = 0                                                        |       |
| if( ( i < lastSubBlock ) && ( i > 0 ) ) {                                        |       |
| <b>coded_sub_block_flag</b> [ xS ][ yS ]                                         | ae(v) |
| inferSbDcSigCoeffFlag = 1                                                        |       |
| }                                                                                |       |
| firstSigScanPosSb = numSbCoeff                                                   |       |
| lastSigScanPosSb = -1                                                            |       |
| remBinsPass1 = ( ( log2SbW + log2SbH ) < 4 ? 8 : 32 )                            |       |
| firstPosMode0 = ( i == lastSubBlock ? lastScanPos : numSbCoeff - 1 )             |       |
| firstPosMode1 = -1                                                               |       |
| for( n = firstPosMode0; n >= 0 && remBinsPass1 >= 4; n-- ) {                     |       |
| xC = ( xS << log2SbW ) + DiagScanOrder[ log2SbW ][ log2SbH ][ n ][ 0 ]           |       |
| yC = ( yS << log2SbH ) + DiagScanOrder[ log2SbW ][ log2SbH ][ n ][ 1 ]           |       |
| if( coded_sub_block_flag[ xS ][ yS ] && ( n > 0    !inferSbDcSigCoeffFlag ) &&   |       |
| ( xC != LastSignificantCoeffX    yC != LastSignificantCoeffY ) ) {               |       |
| <b>sig_coeff_flag</b> [ xC ][ yC ]                                               | ae(v) |
| remBinsPass1--                                                                   |       |
| if( sig_coeff_flag[ xC ][ yC ] )                                                 |       |
| inferSbDcSigCoeffFlag = 0                                                        |       |
| }                                                                                |       |
| if( sig_coeff_flag[ xC ][ yC ] ) {                                               |       |
| if( !transform_skip_flag[ x0 ][ y0 ] ) {                                         |       |
| numSigCoeff++                                                                    |       |

도면3c

|                                                                                                                                                                                      |       |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| if( ( n >= 8 && i == 0 && ( log2TbWidth == 2    log2TbWidth == 3 )<br>&& ( log2TbWidth == log2TbHeight ) )    ( ( i == 1    i == 2 )<br>&& log2TbWidth >= 3 && log2TbHeight >= 3 ) ) |       |
| numZeroOutSigCoeff++                                                                                                                                                                 |       |
| }                                                                                                                                                                                    |       |
| <b>abs_level_gtx_flag[ n ][ 0 ]</b>                                                                                                                                                  | ae(v) |
| remBinsPass1--                                                                                                                                                                       |       |
| if( abs_level_gtx_flag[ n ][ 0 ] ) {                                                                                                                                                 |       |
| <b>par_level_flag[ n ]</b>                                                                                                                                                           | ae(v) |
| remBinsPass1--                                                                                                                                                                       |       |
| <b>abs_level_gtx_flag[ n ][ 1 ]</b>                                                                                                                                                  | ae(v) |
| remBinsPass1--                                                                                                                                                                       |       |
| }                                                                                                                                                                                    |       |
| if( lastSigScanPosSb == -1 )                                                                                                                                                         |       |
| lastSigScanPosSb = n                                                                                                                                                                 |       |
| firstSigScanPosSb = n                                                                                                                                                                |       |
| }                                                                                                                                                                                    |       |
| AbsLevelPass1[ xC ][ yC ] = sig_coeff_flag[ xC ][ yC ] + par_level_flag[ n ] +<br>abs_level_gtx_flag[ n ][ 0 ] + 2 * abs_level_gtx_flag[ n ][ 1 ]                                    |       |
| if( dep_quant_enabled_flag )                                                                                                                                                         |       |
| QState = QStateTransTable[ QState ][ AbsLevelPass1[ xC ][ yC ] & 1 ]                                                                                                                 |       |
| if( remBinsPass1 < 4 )                                                                                                                                                               |       |
| firstPosModel = n - 1                                                                                                                                                                |       |
| }                                                                                                                                                                                    |       |
| for( n = numSbCoeff - 1; n >= firstPosModel; n-- ) {                                                                                                                                 |       |
| xC = ( xS << log2SbW ) + DiagScanOrder[ log2SbW ][ log2SbH ][ n ][ 0 ]                                                                                                               |       |
| yC = ( yS << log2SbH ) + DiagScanOrder[ log2SbW ][ log2SbH ][ n ][ 1 ]                                                                                                               |       |
| if( abs_level_gtx_flag[ n ][ 1 ] )                                                                                                                                                   |       |
| <b>abs_remainder[ n ]</b>                                                                                                                                                            | ae(v) |
| AbsLevel[ xC ][ yC ] = AbsLevelPass1[ xC ][ yC ] + 2 * abs_remainder[ n ]                                                                                                            |       |
| }                                                                                                                                                                                    |       |
| for( n = firstPosModel; n >= 0; n-- ) {                                                                                                                                              |       |
| xC = ( xS << log2SbW ) + DiagScanOrder[ log2SbW ][ log2SbH ][ n ][ 0 ]                                                                                                               |       |
| yC = ( yS << log2SbH ) + DiagScanOrder[ log2SbW ][ log2SbH ][ n ][ 1 ]                                                                                                               |       |
| <b>dec_abs_level[ n ]</b>                                                                                                                                                            | ae(v) |
| if( AbsLevel[ xC ][ yC ] > 0 )                                                                                                                                                       |       |
| firstSigScanPosSb = n                                                                                                                                                                |       |
| if( dep_quant_enabled_flag )                                                                                                                                                         |       |
| QState = QStateTransTable[ QState ][ AbsLevel[ xC ][ yC ] & 1 ]                                                                                                                      |       |
| }                                                                                                                                                                                    |       |
| if( dep_quant_enabled_flag    !sign_data_hiding_enabled_flag )                                                                                                                       |       |
| signHidden = 0                                                                                                                                                                       |       |
| Else                                                                                                                                                                                 |       |

도면3d

|                                                                                                                                                  |       |
|--------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| signHidden = ( lastSigScanPosSb - firstSigScanPosSb > 3 ? 1 : 0 )                                                                                |       |
| for( n = numSbCoeff - 1; n >= 0; n-- ) {                                                                                                         |       |
| xC = ( xS << log2SbW ) + DiagScanOrder[ log2SbW ][ log2SbH ][ n ][ 0 ]                                                                           |       |
| yC = ( yS << log2SbH ) + DiagScanOrder[ log2SbW ][ log2SbH ][ n ][ 1 ]                                                                           |       |
| if( ( AbsLevel[ xC ][ yC ] > 0 ) &&<br>( !signHidden    ( n != firstSigScanPosSb ) ) )                                                           |       |
| coeff_sign_flag[ n ]                                                                                                                             | ae(v) |
| }                                                                                                                                                |       |
| if( dep_quant_enabled_flag ) {                                                                                                                   |       |
| QState = startQStateSb                                                                                                                           |       |
| for( n = numSbCoeff - 1; n >= 0; n-- ) {                                                                                                         |       |
| xC = ( xS << log2SbW ) + DiagScanOrder[ log2SbW ][ log2SbH ][ n ][ 0 ]                                                                           |       |
| yC = ( yS << log2SbH ) + DiagScanOrder[ log2SbW ][ log2SbH ][ n ][ 1 ]                                                                           |       |
| if( AbsLevel[ xC ][ yC ] > 0 )                                                                                                                   |       |
| TransCoeffLevel[ x0 ][ y0 ][ cldx ][ xC ][ yC ] =<br>( 2 * AbsLevel[ xC ][ yC ] - ( QState > 1 ? 1 : 0 ) ) *<br>( 1 - 2 * coeff_sign_flag[ n ] ) |       |
| QState = QStateTransTable[ QState ][ par_level_flag[ n ] ]                                                                                       |       |
| } else {                                                                                                                                         |       |
| sumAbsLevel = 0                                                                                                                                  |       |
| for( n = numSbCoeff - 1; n >= 0; n-- ) {                                                                                                         |       |
| xC = ( xS << log2SbW ) + DiagScanOrder[ log2SbW ][ log2SbH ][ n ][ 0 ]                                                                           |       |
| yC = ( yS << log2SbH ) + DiagScanOrder[ log2SbW ][ log2SbH ][ n ][ 1 ]                                                                           |       |
| if( AbsLevel[ xC ][ yC ] > 0 ) {                                                                                                                 |       |
| TransCoeffLevel[ x0 ][ y0 ][ cldx ][ xC ][ yC ] =<br>AbsLevel[ xC ][ yC ] * ( 1 - 2 * coeff_sign_flag[ n ] )                                     |       |
| if( signHidden ) {                                                                                                                               |       |
| sumAbsLevel += AbsLevel[ xC ][ yC ]                                                                                                              |       |
| if( ( n == firstSigScanPosSb ) && ( sumAbsLevel % 2 ) == 1 )                                                                                     |       |
| TransCoeffLevel[ x0 ][ y0 ][ cldx ][ xC ][ yC ] =<br>-TransCoeffLevel[ x0 ][ y0 ][ cldx ][ xC ][ yC ]                                            |       |
| }                                                                                                                                                |       |
| }                                                                                                                                                |       |
| }                                                                                                                                                |       |
| }                                                                                                                                                |       |
| }                                                                                                                                                |       |
| }                                                                                                                                                |       |

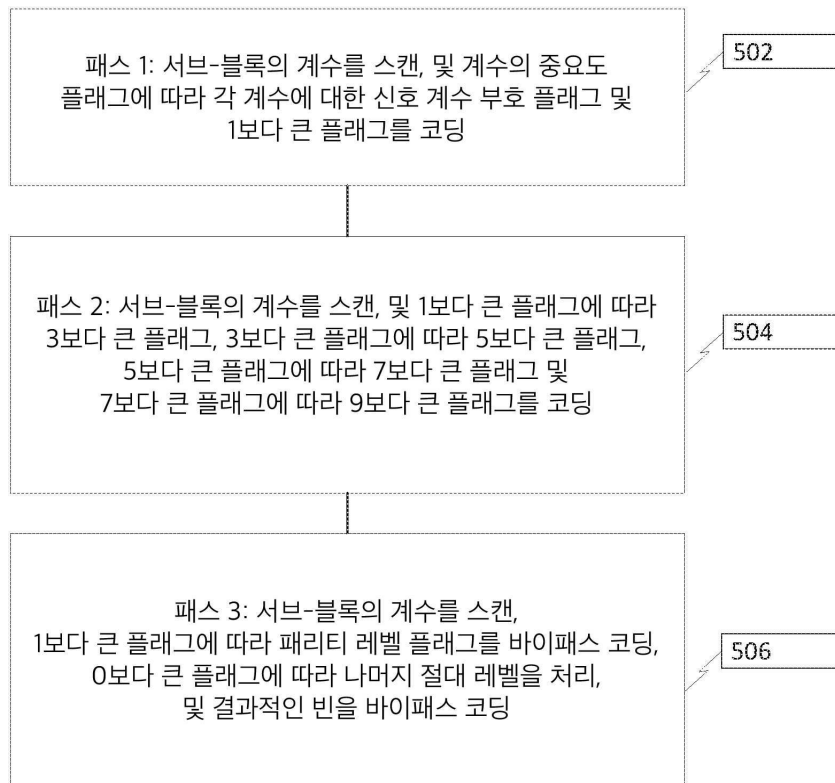
도면4a

|                                                                                       |       |
|---------------------------------------------------------------------------------------|-------|
| residual_ts_coding( x0, y0, log2TbWidth, log2TbHeight, cIdx ) {                       | 설명자   |
| log2SbSize = ( Min( log2TbWidth, log2TbHeight ) < 2 ? 1 : 2 )                         |       |
| numSbCoeff = 1 << ( log2SbSize << 1 )                                                 |       |
| lastSubBlock = ( 1 << ( log2TbWidth + log2TbHeight - 2 * log2SbSize ) ) - 1           |       |
| inferSbCbf = 1                                                                        |       |
| MaxCcbs = 2 * ( 1 << log2TbWidth ) * ( 1 << log2TbHeight )                            |       |
| for( i = 0; i <= lastSubBlock; i++ ) {                                                |       |
| xS = DiagScanOrder[ log2TbWidth - log2SbSize ][ log2TbHeight - log2SbSize ][ i ][ 0 ] |       |
| yS = DiagScanOrder[ log2TbWidth - log2SbSize ][ log2TbHeight - log2SbSize ][ i ][ 1 ] |       |
| if( ( i != lastSubBlock    !inferSbCbf ) {                                            |       |
| <b>coded_sub_block_flag</b> [ xS ][ yS ]                                              | ae(v) |
| MaxCcbs--                                                                             |       |
| }                                                                                     |       |
| if( coded_sub_block_flag[ xS ][ yS ] && i < lastSubBlock )                            |       |
| inferSbCbf = 0                                                                        |       |
| /* 제1 스캔 패스 */                                                                        |       |
| inferSbSigCoeffFlag = 1                                                               |       |
| for( n = 0; n <= numSbCoeff - 1; n++ ) {                                              |       |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]       |       |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]       |       |
| if( coded_sub_block_flag[ xS ][ yS ] &&                                               |       |
| ( n != numSbCoeff - 1    !inferSbSigCoeffFlag ) ) {                                   |       |
| <b>sig_coeff_flag</b> [ xC ][ yC ]                                                    | ae(v) |
| MaxCcbs--                                                                             |       |
| if( sig_coeff_flag[ xC ][ yC ] )                                                      |       |
| inferSbSigCoeffFlag = 0                                                               |       |
| }                                                                                     |       |
| if( sig_coeff_flag[ xC ][ yC ] {                                                      |       |
| <b>coeff_sign_flag</b> [ n ]                                                          | ae(v) |
| MaxCcbs--                                                                             |       |
| <b>abs_level_gtx_flag</b> [ n ][ 0 ]                                                  | ae(v) |
| MaxCcbs--                                                                             |       |
| if( abs_level_gtx_flag[ n ][ 0 ] ) {                                                  |       |
| <b>par_level_flag</b> [ n ]                                                           | ae(v) |
| MaxCcbs--                                                                             |       |
| }                                                                                     |       |
| }                                                                                     |       |
| AbsLevelPassX[ xC ][ yC ] =                                                           |       |
| sig_coeff_flag[ xC ][ yC ] + par_level_flag[ n ] + abs_level_gtx_flag[ n ][ 0 ]       |       |

도면4b

|                                                                                                                                            |       |
|--------------------------------------------------------------------------------------------------------------------------------------------|-------|
| }                                                                                                                                          |       |
| /* X보다 큰 스캔 패스(numGtXFlags=5) */                                                                                                           |       |
| for( j = 1; j < 5; j++ ) {                                                                                                                 |       |
| for( n = 0; n <= numSbCoeff - 1; n++ ) {                                                                                                   |       |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]                                                            |       |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]                                                            |       |
| if( abs_level_gtx_flag[ n ][ j - 1 ] )                                                                                                     |       |
| abs_level_gtx_flag[ n ][ j ]                                                                                                               | ae(v) |
| MaxCcbs--                                                                                                                                  |       |
| AbsLevelPassX[ xC ][ yC ] += 2 * abs_level_gtx_flag[ n ][ j ]                                                                              |       |
| }                                                                                                                                          |       |
| }                                                                                                                                          |       |
| /* 나머지 스캔 패스 */                                                                                                                            |       |
| for( n = 0; n <= numSbCoeff - 1; n++ ) {                                                                                                   |       |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]                                                            |       |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]                                                            |       |
| if( abs_level_gtx_flag[ n ][ 4 ] )                                                                                                         |       |
| abs_remainder[ n ]                                                                                                                         | ae(v) |
| TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 - 2 * coeff_sign_flag[ n ] ) *<br>( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] ) |       |
| }                                                                                                                                          |       |
| }                                                                                                                                          |       |
| }                                                                                                                                          |       |

도면5



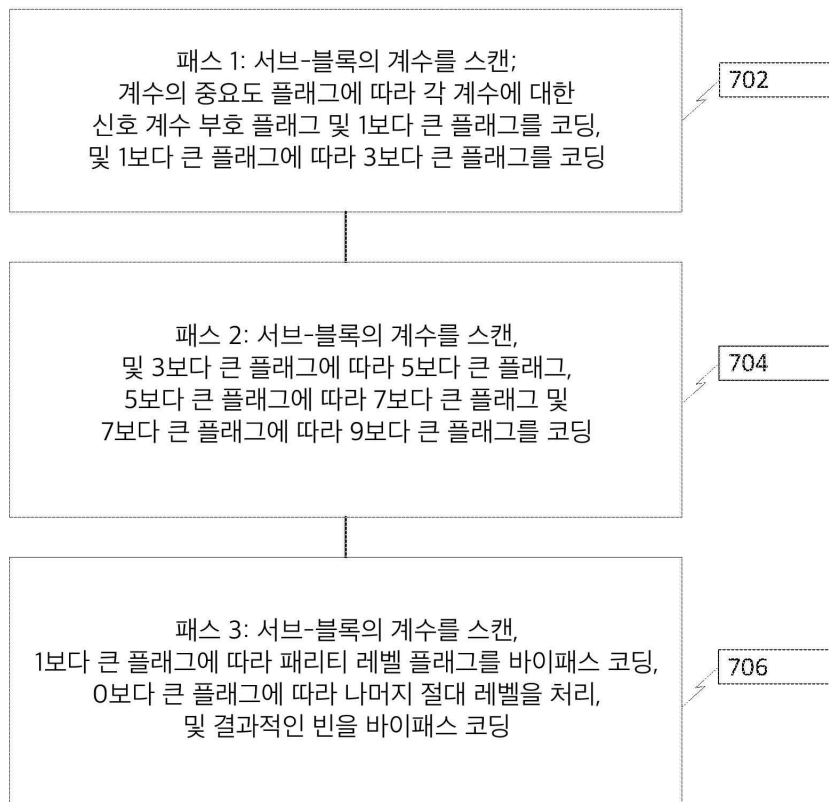
도면6a

|                                                                                             |       |
|---------------------------------------------------------------------------------------------|-------|
| residual_ts_coding( x0, y0, log2TbWidth, log2TbHeight, cldx ) {                             | 설명자   |
| log2SbSize = ( Min( log2TbWidth, log2TbHeight ) < 2 ? 1 : 2 )                               |       |
| numSbCoeff = 1 << ( log2SbSize << 1 )                                                       |       |
| lastSubBlock = ( 1 << ( log2TbWidth + log2TbHeight - 2 * log2SbSize ) ) - 1                 |       |
| inferSbCbf = 1                                                                              |       |
| MaxCcbs = 2 * ( 1 << log2TbWidth ) * ( 1 << log2TbHeight )                                  |       |
| for( i=0; i <= lastSubBlock; i++ ) {                                                        |       |
| xS = DiagScanOrder[ log2TbWidth - log2SbSize ][ log2TbHeight - log2SbSize ][ i ][ 0 ]       |       |
| yS = DiagScanOrder[ log2TbWidth - log2SbSize ][ log2TbHeight - log2SbSize ][ i ][ 1 ]       |       |
| if( ( i != lastSubBlock    !inferSbCbf ) {                                                  |       |
| <b>coded_sub_block_flag</b> [ xS ][ yS ]                                                    | ae(v) |
| MaxCcbs--                                                                                   |       |
| }                                                                                           |       |
| if( coded_sub_block_flag[ xS ][ yS ] && i < lastSubBlock )                                  |       |
| inferSbCbf = 0                                                                              |       |
| /* 제1 스캔 패스 */                                                                              |       |
| inferSbSigCoeffFlag = 1                                                                     |       |
| for( n = 0; n <= numSbCoeff - 1; n++ ) {                                                    |       |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]             |       |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]             |       |
| if( coded_sub_block_flag[ xS ][ yS ] && ( n != numSbCoeff - 1    !inferSbSigCoeffFlag ) ) { |       |
| <b>sig_coeff_flag</b> [ xC ][ yC ]                                                          | ae(v) |
| MaxCcbs--                                                                                   |       |
| if( sig_coeff_flag[ xC ][ yC ] )                                                            |       |
| inferSbSigCoeffFlag = 0                                                                     |       |
| }                                                                                           |       |
| if( sig_coeff_flag[ xC ][ yC ] {                                                            |       |
| <b>coeff_sign_flag</b> [ n ]                                                                | ae(v) |
| MaxCcbs--                                                                                   |       |
| <b>abs_level_gtx_flag</b> [ n ][ 0 ]                                                        | ae(v) |
| MaxCcbs--                                                                                   |       |
| }                                                                                           |       |
| AbsLevelPassX[ xC ][ yC ] =                                                                 |       |
| sig_coeff_flag[ xC ][ yC ] + abs_level_gtx_flag[ n ][ 0 ]                                   |       |
| }                                                                                           |       |
| /* X보다 큰 스캔 패스(numGtXFlags=5) */                                                            |       |
| for( n = 0; n <= numSbCoeff - 1; n++ ) {                                                    |       |

도면6b

|                                                                                                                                              |       |
|----------------------------------------------------------------------------------------------------------------------------------------------|-------|
| $xC = (xS \ll \log 2SbSize) + DiagScanOrder[\log 2SbSize][\log 2SbSize][n][0]$                                                               |       |
| $yC = (yS \ll \log 2SbSize) + DiagScanOrder[\log 2SbSize][\log 2SbSize][n][1]$                                                               |       |
| for( $j = 1; j < 5; j++$ ) {                                                                                                                 |       |
| if( $abs\_level\_gtx\_flag[n][j-1]$ )                                                                                                        |       |
| $abs\_level\_gtx\_flag[n][j]$                                                                                                                | ae(v) |
| MaxCbs--                                                                                                                                     |       |
| $AbsLevelPassX[xC][yC] += 2 * abs\_level\_gtx\_flag[n][j]$                                                                                   |       |
| }                                                                                                                                            |       |
| /* 나머지 스캔 패스 */                                                                                                                              |       |
| for( $n = 0; n \leq numSbCoeff - 1; n++$ ) {                                                                                                 |       |
| $xC = (xS \ll \log 2SbSize) + DiagScanOrder[\log 2SbSize][\log 2SbSize][n][0]$                                                               |       |
| $yC = (yS \ll \log 2SbSize) + DiagScanOrder[\log 2SbSize][\log 2SbSize][n][1]$                                                               |       |
| if( $abs\_level\_gtx\_flag[n][0]$ )                                                                                                          |       |
| $par\_level\_flag[n]$                                                                                                                        | ae(v) |
| if( $abs\_level\_gtx\_flag[n][4]$ )                                                                                                          |       |
| $abs\_remainder[n]$                                                                                                                          | ae(v) |
| $TransCoeffLevel[x0][y0][cIdx][xC][yC] = (1 - 2 * coeff\_sign\_flag[n]) * (AbsLevelPassX[xC][yC] + par\_level\_flag[n] + abs\_remainder[n])$ |       |
| }                                                                                                                                            |       |
| }                                                                                                                                            |       |

도면7



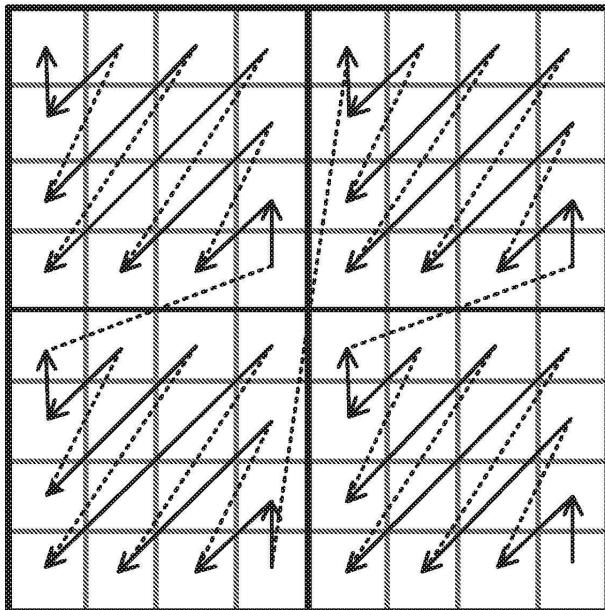
도면8a

|                                                                                       |       |
|---------------------------------------------------------------------------------------|-------|
| residual_ts_coding( x0, y0, log2TbWidth, log2TbHeight, cIdx ) {                       | 설명자   |
| log2SbSize = ( Min( log2TbWidth, log2TbHeight ) < 2 ? 1 : 2 )                         |       |
| numSbCoeff = 1 << ( log2SbSize << 1 )                                                 |       |
| lastSubBlock = ( 1 << ( log2TbWidth + log2TbHeight - 2 * log2SbSize ) ) - 1           |       |
| inferSbCbf = 1                                                                        |       |
| MaxCcbcs = 2 * ( 1 << log2TbWidth ) * ( 1 << log2TbHeight )                           |       |
| for( i=0; i <= lastSubBlock; i++ ) {                                                  |       |
| xS = DiagScanOrder[ log2TbWidth - log2SbSize ][ log2TbHeight - log2SbSize ][ i ][ 0 ] |       |
| yS = DiagScanOrder[ log2TbWidth - log2SbSize ][ log2TbHeight - log2SbSize ][ i ][ 1 ] |       |
| if( ( i != lastSubBlock    !inferSbCbf ) {                                            |       |
| <b>coded_sub_block_flag</b> [ xS ][ yS ]                                              | ae(v) |
| MaxCcbcs--                                                                            |       |
| }                                                                                     |       |
| if( coded_sub_block_flag[ xS ][ yS ] && i < lastSubBlock )                            |       |
| inferSbCbf = 0                                                                        |       |
| /* 제1 스캔 패스 */                                                                        |       |
| inferSbSigCoeffFlag = 1                                                               |       |
| for( n = 0; n <= numSbCoeff - 1; n++ ) {                                              |       |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]       |       |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]       |       |
| if( coded_sub_block_flag[ xS ][ yS ] &&                                               |       |
| ( n != numSbCoeff - 1    !inferSbSigCoeffFlag ) ) {                                   |       |
| <b>sig_coeff_flag</b> [ xC ][ yC ]                                                    | ae(v) |
| MaxCcbcs--                                                                            |       |
| if( sig_coeff_flag[ xC ][ yC ] )                                                      |       |
| inferSbSigCoeffFlag = 0                                                               |       |
| }                                                                                     |       |
| if( sig_coeff_flag[ xC ][ yC ] {                                                      |       |
| <b>coeff_sign_flag</b> [ n ]                                                          | ae(v) |
| MaxCcbcs--                                                                            |       |
| <b>abs_level_gtx_flag</b> [ n ][ 0 ]                                                  | ae(v) |
| MaxCcbcs--                                                                            |       |
| if( abs_level_gtx_flag[ n ][ 0 ] ) {                                                  |       |
| <b>abs_level_gtx_flag</b> [ n ][ 1 ]                                                  | ae(v) |
| MaxCcbcs--                                                                            |       |
| }                                                                                     |       |
| }                                                                                     |       |

도면8b

|                                                                                                |         |
|------------------------------------------------------------------------------------------------|---------|
| $AbsLevelPassX[ xC ][ yC ] =$                                                                  |         |
| $\quad sig\_coeff\_flag[ xC ][ yC ] + abs\_level\_gtx\_flag[ n ][ 0 ] + 2 *$                   |         |
| $abs\_level\_gtx\_flag[ n ][ 1 ]$                                                              |         |
| $]$                                                                                            |         |
| $/* \times$ 보다 큰 스캔 패스 (numGtXFlags=5) */                                                      |         |
| $for( n = 0; n <= numSbCoeff - 1; n++ ) \{$                                                    |         |
| $\quad xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]$        |         |
| $\quad yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]$        |         |
| $\quad for( j = 2; j < 5; j++ ) \{$                                                            |         |
| $\quad \quad if( abs\_level\_gtx\_flag[ n ][ j - 1 ] )$                                        |         |
| $\quad \quad \quad abs\_level\_gtx\_flag[ n ][ j ]$                                            | $ae(v)$ |
| $\quad \quad \quad MaxCbs--$                                                                   |         |
| $\quad \quad AbsLevelPassX[ xC ][ yC ] += 2 * abs\_level\_gtx\_flag[ n ][ j ]$                 |         |
| $\quad \quad }$                                                                                |         |
| $\quad }$                                                                                      |         |
| $/* 나머지 스캔 패스 */$                                                                              |         |
| $for( n = 0; n <= numSbCoeff - 1; n++ ) \{$                                                    |         |
| $\quad xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]$        |         |
| $\quad yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]$        |         |
| $\quad if( abs\_level\_gtx\_flag[ n ][ 0 ] )$                                                  |         |
| $\quad \quad par\_level\_flag[ n ]$                                                            | $ae(v)$ |
| $\quad if( abs\_level\_gtx\_flag[ n ][ 4 ] )$                                                  |         |
| $\quad \quad abs\_remainder[ n ]$                                                              | $ae(v)$ |
| $\quad TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 - 2 * coeff\_sign\_flag[ n ] ) *$ |         |
| $\quad \quad ( AbsLevelPassX[ xC ][ yC ] + par\_level\_flag[ n ] +$                            |         |
| $\quad \quad abs\_remainder[ n ] )$                                                            |         |
| $\quad }$                                                                                      |         |
| $\quad }$                                                                                      |         |
| $\}$                                                                                           |         |

도면9



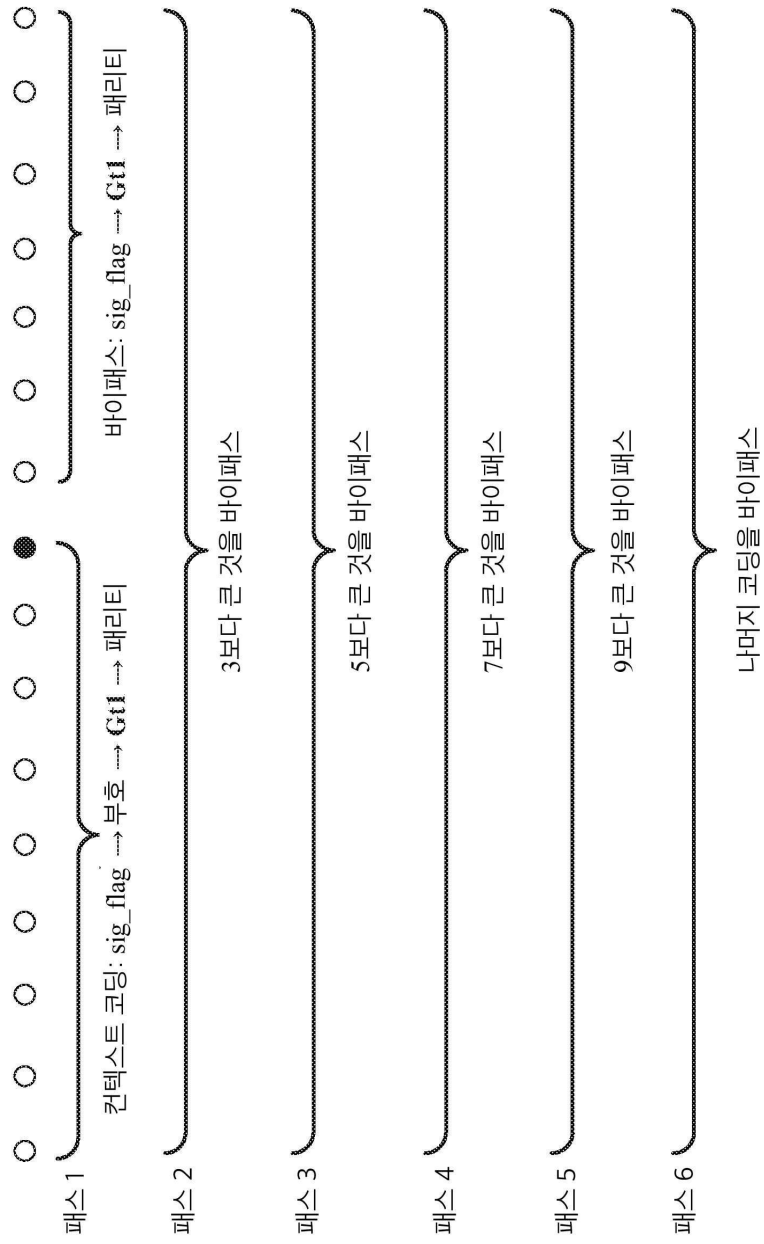
도면10a

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  |
| 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |

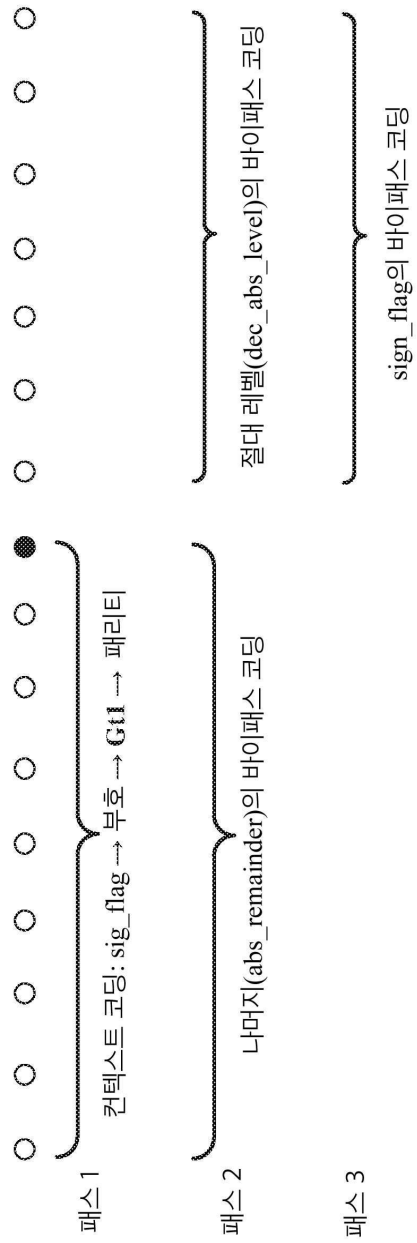
도면10b

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 |
| 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 |
| 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 |
| 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |

도면11



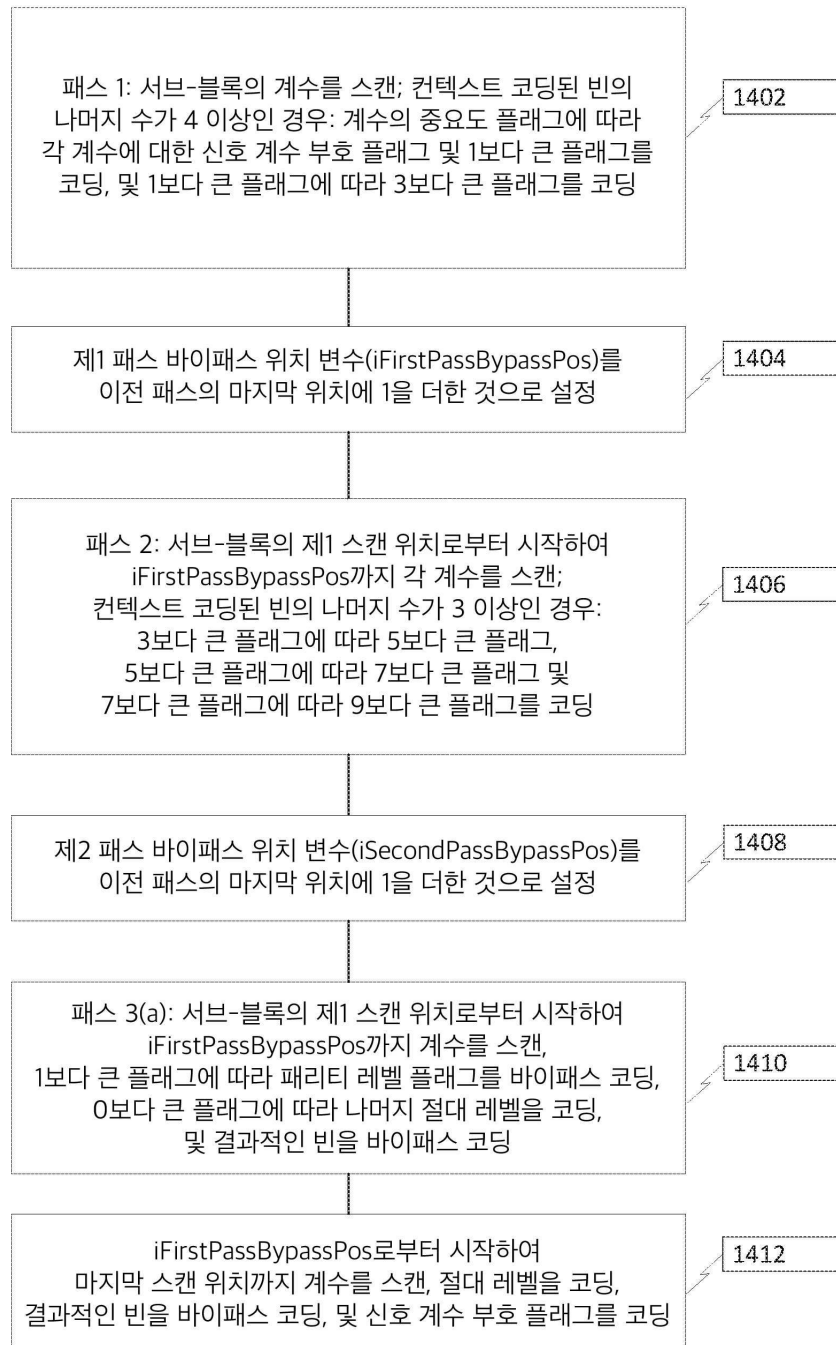
도면12



도면13

|            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| locSumAbs  | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
| cRiceParam | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 1  |
| locSumAbs  | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| cRiceParam | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 2  | 2  | 2  | 2  | 2  | 2  | 2  |

도면14



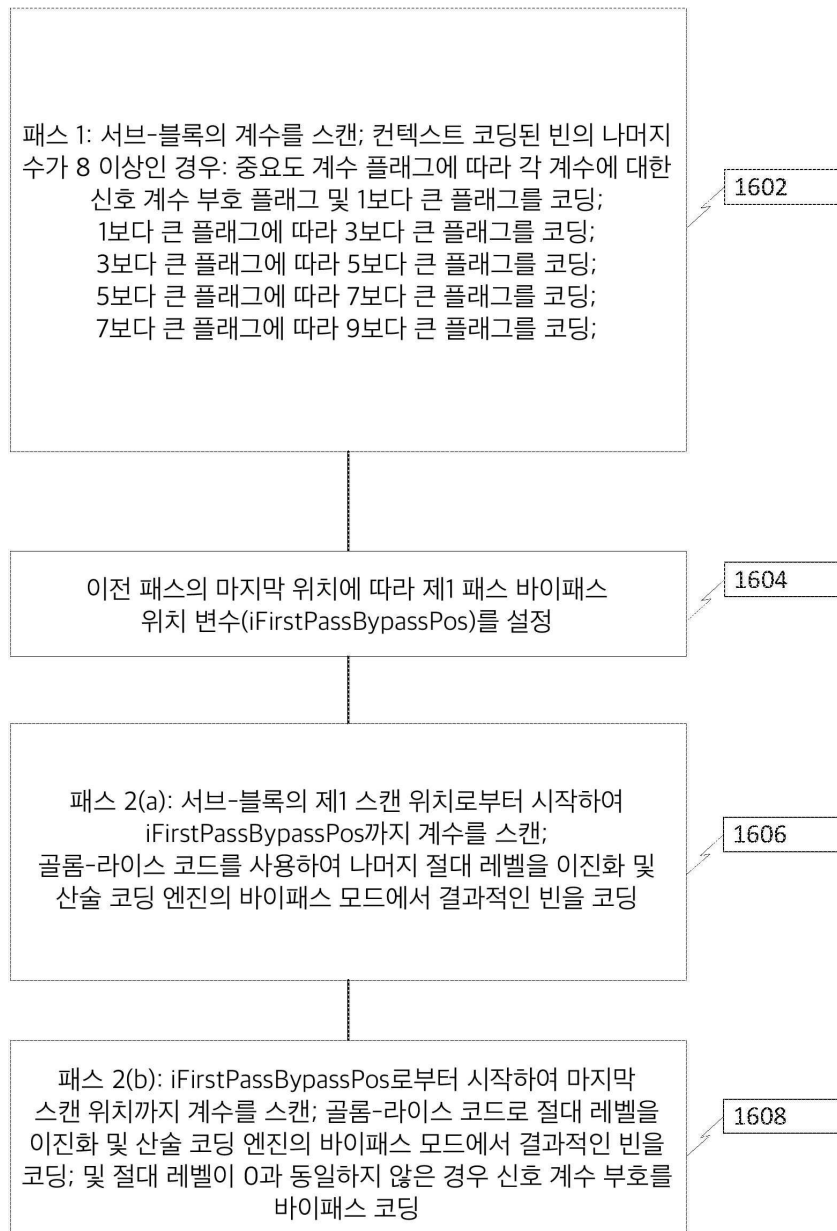
도면15a

|                                                                                       |       |
|---------------------------------------------------------------------------------------|-------|
| residual_ts_coding( x0, y0, log2TbWidth, log2TbHeight, cldx ) {                       | 설명자   |
| log2SbSize = ( Min( log2TbWidth, log2TbHeight ) < 2 ? 1 : 2 )                         |       |
| numSbCoeff = 1 << ( log2SbSize << 1 )                                                 |       |
| lastSubBlock = ( 1 << ( log2TbWidth + log2TbHeight - 2 * log2SbSize ) ) - 1           |       |
| inferSbCbf = 1                                                                        |       |
| MaxCcbs = 2 * ( 1 << log2TbWidth ) * ( 1 << log2TbHeight )                            |       |
| for( i = 0; i <= lastSubBlock; i++ ) {                                                |       |
| xS = DiagScanOrder[ log2TbWidth - log2SbSize ][ log2TbHeight - log2SbSize ][ i ][ 0 ] |       |
| yS = DiagScanOrder[ log2TbWidth - log2SbSize ][ log2TbHeight - log2SbSize ][ i ][ 1 ] |       |
| if( ( i != lastSubBlock    !inferSbCbf ) {                                            |       |
| <b>coded_sub_block_flag</b> [ xS ][ yS ]                                              | ae(v) |
| MaxCcbs--                                                                             |       |
| }                                                                                     |       |
| if( coded_sub_block_flag[ xS ][ yS ] && i < lastSubBlock )                            |       |
| inferSbCbf = 0                                                                        |       |
| /* 제1 스캔 패스 */                                                                        |       |
| inferSbSigCoeffFlag = 1                                                               |       |
| for( n = 0; n <= numSbCoeff - 1 && MaxCcbs >= 4; n++ ) {                              |       |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]       |       |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]       |       |
| if( coded_sub_block_flag[ xS ][ yS ] &&                                               |       |
| ( n != numSbCoeff - 1    !inferSbSigCoeffFlag ) ) {                                   |       |
| <b>sig_coeff_flag</b> [ xC ][ yC ]                                                    | ae(v) |
| MaxCcbs--                                                                             |       |
| if( sig_coeff_flag[ xC ][ yC ] )                                                      |       |
| inferSbSigCoeffFlag = 0                                                               |       |
| }                                                                                     |       |
| if( sig_coeff_flag[ xC ][ yC ] {                                                      |       |
| <b>coeff_sign_flag</b> [ n ]                                                          | ae(v) |
| MaxCcbs--                                                                             |       |
| <b>abs_level_gtx_flag</b> [ n ][ 0 ]                                                  | ae(v) |
| MaxCcbs--                                                                             |       |
| if( abs_level_gtx_flag[ n ][ 0 ] ) {                                                  |       |
| <b>abs_level_gtx_flag</b> [ n ][ 1 ]                                                  | ae(v) |
| MaxCcbs--                                                                             |       |
| }                                                                                     |       |
| }                                                                                     |       |
| }                                                                                     |       |

도면15b

|                                                                                                                                                                     |       |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| AbsLevelPassX[ xC ][ yC ] =<br>sig_coeff_flag[ xC ][ yC ] + abs_level_gtx_flag[ n ][ 0 ] + 2 *<br>abs_level_gtx_flag[ n ][ 1 ]                                      |       |
| }                                                                                                                                                                   |       |
| iFirstPassBypassPos = n;                                                                                                                                            |       |
| /* 제2 스캔 패스 */                                                                                                                                                      |       |
| for( n = 0; n < iFirstPassBypassPos && MaxCbs >= 3; n++ ) {                                                                                                         |       |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]                                                                                     |       |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]                                                                                     |       |
| for( j = 2; j < 5; j++ ) {                                                                                                                                          |       |
| if( abs_level_gtx_flag[ n ][ j - 1 ] )                                                                                                                              |       |
| abs_level_gtx_flag[ n ][ j ]                                                                                                                                        | ae(v) |
| MaxCbs--                                                                                                                                                            |       |
| AbsLevelPassX[ xC ][ yC ] += 2 * abs_level_gtx_flag[ n ][ j ]                                                                                                       |       |
| }                                                                                                                                                                   |       |
| }                                                                                                                                                                   |       |
| iSecondPassBypassPos = n;                                                                                                                                           |       |
| /* 나머지 스캔 패스 */                                                                                                                                                     |       |
| for( n = 0; n < iFirstPassBypassPos ; n++ ) {                                                                                                                       |       |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]                                                                                     |       |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]                                                                                     |       |
| minLevel = ( scanPos >= iSecondPassBypassPos ? 4 : 10);                                                                                                             |       |
| if( abs_level_gtx_flag[ n ][ 0 ] )                                                                                                                                  |       |
| par_level_flag[ n ]                                                                                                                                                 | ae(v) |
| if( abs_level_gtx_flag[ n ][ 4 ] )                                                                                                                                  |       |
| abs_remainder[ n ]                                                                                                                                                  | ae(v) |
| TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 - 2 * coeff_sign_flag[ n ] ) *<br>( AbsLevelPassX[ xC ][ yC ] + par_level_flag[ n ] +<br>abs_remainder[ n ] ) |       |
| }                                                                                                                                                                   |       |
| /* dec_abs_level 스캔 패스 */                                                                                                                                           |       |
| for( n = iFirstPassBypassPos; n <= numSbCoeff - 1; n++ ) {                                                                                                          |       |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]                                                                                     |       |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]                                                                                     |       |
| dec_abs_level[ n ]                                                                                                                                                  | ae(v) |
| if( sig_coeff_flag[ xC ][ yC ] )                                                                                                                                    |       |
| coeff_sign_flag[ n ]                                                                                                                                                | ae(v) |
| TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 - 2 * coeff_sign_flag[ n ] ) *<br>dec_abs_level[n]                                                            |       |
| }                                                                                                                                                                   |       |
| }                                                                                                                                                                   |       |
| }                                                                                                                                                                   |       |

도면16



도면17a

|                                                                                       |       |
|---------------------------------------------------------------------------------------|-------|
| residual_ts_coding( x0, y0, log2TbWidth, log2TbHeight, cldx ) {                       | 설명자   |
| log2SbSize = ( Min( log2TbWidth, log2TbHeight ) < 2 ? 1 : 2 )                         |       |
| numSbCoeff = 1 << ( log2SbSize << 1 )                                                 |       |
| lastSubBlock = ( 1 << ( log2TbWidth + log2TbHeight - 2 * log2SbSize ) ) - 1           |       |
| inferSbCbf = 1                                                                        |       |
| MaxCcbs = 2 * ( 1 << log2TbWidth ) * ( 1 << log2TbHeight )                            |       |
| for( i = 0; i <= lastSubBlock; i++ ) {                                                |       |
| xS = DiagScanOrder[ log2TbWidth - log2SbSize ][ log2TbHeight - log2SbSize ][ i ][ 0 ] |       |
| yS = DiagScanOrder[ log2TbWidth - log2SbSize ][ log2TbHeight - log2SbSize ][ i ][ 1 ] |       |
| if( ( i != lastSubBlock    !inferSbCbf ) {                                            |       |
| <b>coded_sub_block_flag</b> [ xS ][ yS ]                                              | ae(v) |
| }                                                                                     |       |
| if( coded_sub_block_flag[ xS ][ yS ] && i < lastSubBlock )                            |       |
| inferSbCbf = 0                                                                        |       |
| /* 제1 스캔 패스 */                                                                        |       |
| inferSbSigCoeffFlag = 1                                                               |       |
| for( n = 0; n <= numSbCoeff - 1 && MaxCcbs >= 8; n++ ) {                              |       |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]       |       |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]       |       |
| if( coded_sub_block_flag[ xS ][ yS ] &&                                               |       |
| ( n != numSbCoeff - 1    !inferSbSigCoeffFlag ) ) {                                   |       |
| <b>sig_coeff_flag</b> [ xC ][ yC ]                                                    | ae(v) |
| MaxCcbs--                                                                             |       |
| if( sig_coeff_flag[ xC ][ yC ] )                                                      |       |
| inferSbSigCoeffFlag = 0                                                               |       |
| }                                                                                     |       |
| CoeffSignLevel[ xC ][ yC ] = 0                                                        |       |
| if( sig_coeff_flag[ xC ][ yC ] {                                                      |       |
| <b>coeff_sign_flag</b> [ n ]                                                          | ae(v) |
| MaxCcbs--                                                                             |       |
| CoeffSignLevel[ xC ][ yC ] = ( coeff_sign_flag[ n ] > 0 ? -1 : 1 )                    |       |
| <b>abs_level_gtx_flag</b> [ n ][ 0 ]                                                  | ae(v) |
| MaxCcbs--                                                                             |       |
| if( abs_level_gtx_flag[ n ][ 0 ] ) {                                                  |       |
| <b>par_level_flag</b> [ n ]                                                           | ae(v) |
| MaxCcbs--                                                                             |       |
| }                                                                                     |       |
| }                                                                                     |       |
| }                                                                                     |       |
| AbsLevelPassX[ xC ][ yC ] =                                                           |       |
| sig_coeff_flag[ xC ][ yC ] + par_level_flag[ n ] + abs_level_gtx_flag[ n ][ 0 ]       |       |

도면17b

|                                                                                      |       |
|--------------------------------------------------------------------------------------|-------|
| —+—                                                                                  |       |
| /* X보다 큰 스캔 패스 (numGtXFlags=5) */                                                    |       |
| for( n = 0; n <= numSbCoeff - 1; n++ ) {                                             |       |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]      |       |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]      |       |
| for( j = 1; j < 5; j++ ) {                                                           |       |
| if( abs_level_gtx_flag[ n ][ j - 1 ] ) {                                             |       |
| abs_level_gtx_flag[ n ][ j ]                                                         | ae(v) |
| MaxCbs--                                                                             |       |
| AbsLevelPassX[ xC ][ yC ] += 2 * abs_level_gtx_flag[ n ][ j ]                        |       |
| }                                                                                    |       |
| }                                                                                    |       |
| }                                                                                    |       |
| iFirstPassBypassPos = n;                                                             |       |
| /* 나머지 스캔 패스 */                                                                      |       |
| for( n = 0; n <= numSbCoeff - 1 - iFirstPassBypassPos; n++ ) {                       |       |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]      |       |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]      |       |
| if( abs_level_gtx_flag[ n ][ 4 ] )                                                   |       |
| abs_remainder[ n ]                                                                   | ae(v) |
| if( intra_bdpcn_flag == 0 ) {                                                        |       |
| absRightCoeff = abs( TransCoeffLevel[ x0 ][ y0 ][ cldx ][ xC - 1 ][ yC ] )           |       |
| absBelowCoeff = abs( TransCoeffLevel[ x0 ][ y0 ][ cldx ][ xC ][ yC - 1 ] )           |       |
| predCoeff = Max( absRightCoeff, absBelowCoeff )                                      |       |
| if( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] == 1 && predCoeff > 0 )           |       |
| TransCoeffLevel[ x0 ][ y0 ][ cldx ][ xC ][ yC ] =                                    |       |
| ( 1 - 2 * coeff_sign_flag[ n ] ) * predCoeff                                         |       |
| else if( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] <= predCoeff )               |       |
| TransCoeffLevel[ x0 ][ y0 ][ cldx ][ xC ][ yC ] = ( 1 - 2 * coeff_sign_flag[ n ] ) * |       |
| ( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] - 1 )                               |       |
| Else                                                                                 |       |
| TransCoeffLevel[ x0 ][ y0 ][ cldx ][ xC ][ yC ] = ( 1 - 2 * coeff_sign_flag[ n ] ) * |       |
| ( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] )                                   |       |
| } else                                                                               |       |
| TransCoeffLevel[ x0 ][ y0 ][ cldx ][ xC ][ yC ] = ( 1 - 2 * coeff_sign_flag[ n ] ) * |       |
| ( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] )                                   |       |
| }                                                                                    |       |
| /* dec_abs_level 스캔 패스 */                                                            |       |
| for( n = iFirstPassBypassPos; n <= numSbCoeff - 1; n++ ) {                           |       |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]      |       |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]      |       |

도면17c

|                                                                                                                                                              |              |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| <i>dec_abs_level[ n ]</i>                                                                                                                                    | <i>ae(v)</i> |
| <i>if( dec_abs_level[ n ] )</i>                                                                                                                              |              |
| <i>coeff_sign_flag[ n ]</i>                                                                                                                                  | <i>ae(v)</i> |
| <i>if( intra_bdpcm_flag == 0 ) {</i>                                                                                                                         |              |
| <i>absRightCoeff = abs( TransCoeffLevel[ x0 ][ y0 ][ cldx ][ xC - 1 ][ yC ] )</i>                                                                            |              |
| <i>absBelowCoeff = abs( TransCoeffLevel[ x0 ][ y0 ][ cldx ][ xC ][ yC - 1 ] )</i>                                                                            |              |
| <i>predCoeff = Max( absRightCoeff, absBelowCoeff )</i>                                                                                                       |              |
| <i>if( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] == 1 &amp;&amp; predCoeff &gt; 0 )</i>                                                                 |              |
| <i>TransCoeffLevel[ x0 ][ y0 ][ cldx ][ xC ][ yC ] =</i><br><i>( 1 - 2 * coeff_sign_flag[ n ] ) * predCoeff</i>                                              |              |
| <i>else if( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] &lt;= predCoeff )</i>                                                                             |              |
| <i>TransCoeffLevel[ x0 ][ y0 ][ cldx ][ xC ][ yC ] = ( 1 - 2 * coeff_sign_flag[ n ] ) *</i><br><i>( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] - 1 )</i> |              |
| <i>Else</i>                                                                                                                                                  |              |
| <i>TransCoeffLevel[ x0 ][ y0 ][ cldx ][ xC ][ yC ] = ( 1 - 2 * coeff_sign_flag[ n ] ) *</i><br><i>( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] )</i>     |              |
| <i>} else</i>                                                                                                                                                |              |
| <i>TransCoeffLevel[ x0 ][ y0 ][ cldx ][ xC ][ yC ] = ( 1 - 2 * coeff_sign_flag[ n ] ) *</i><br><i>( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] )</i>     |              |
| <i>/</i>                                                                                                                                                     |              |
| <i>}</i>                                                                                                                                                     |              |
| <i>}</i>                                                                                                                                                     |              |

도면18

|                   |           |           |           |           |           |           |           |           |           |           |           |           |           |           |           |           |
|-------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| <b>locSumAbs</b>  | <b>0</b>  | <b>1</b>  | <b>2</b>  | <b>3</b>  | <b>4</b>  | <b>5</b>  | <b>6</b>  | <b>7</b>  | <b>8</b>  | <b>9</b>  | <b>10</b> | <b>11</b> | <b>12</b> | <b>13</b> | <b>14</b> | <b>15</b> |
| <b>cRiceParam</b> | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 1         | 1         | 1         | 1         | 1         | 1         | 1         | 2         | 2         |
| <b>locSumAbs</b>  | <b>16</b> | <b>17</b> | <b>18</b> | <b>19</b> | <b>20</b> | <b>21</b> | <b>22</b> | <b>23</b> | <b>24</b> | <b>25</b> | <b>26</b> | <b>27</b> | <b>28</b> | <b>29</b> | <b>30</b> | <b>31</b> |
| <b>cRiceParam</b> | 2         | 2         | 2         | 2         | 2         | 2         | 2         | 2         | 2         | 2         | 2         | 2         | 3         | 3         | 3         | 3         |