(12) 특허협력조약에 의하여 공개된 국제출원

(19) 세계지식재산권기구
국제사무국

(43) 국제공개일
2020 년 9 월 24 일 (24.09.2020)

WIPO | PCT

(10) 국제공개번호
WO 2020/189971 A1

(51) 국제특허분류:
H04N 19/61 (2014.01) *H04N 19/137* (2014.01)
H04N 19/11 (2014.01) *H04N 19/70* (2014.01)
H04N 19/593 (2014.01) *H04N 19/176* (2014.01)
H04N 19/132 (2014.01)

(21) 국제출원번호: PCT/KR2020/003515

(22) 국제출원일: 2020 년 3 월 13 일 (13.03.2020)

(25) 출원언어: 한국어

(26) 공개언어: 한국어

(30) 우선권정보:
62/818,757 2019 년 3 월 15 일 (15.03.2019) US

(71) 출원인: 엘지전자 주식회사 (**LG ELECTRONICS INC.**) [KR/KR]; 07336 서울시 영등포구 여의대로 128, Seoul (KR).

(72) 발명자: **유선미 (YOO, Sunmi)**; 06772 서울시 서초구 양재대로11길 19 LG전자 특허센터, Seoul (KR). **남정학 (NAM, Junghak)**; 06772 서울시 서초구 양재대로11길 19 LG전자 특허센터, Seoul (KR).

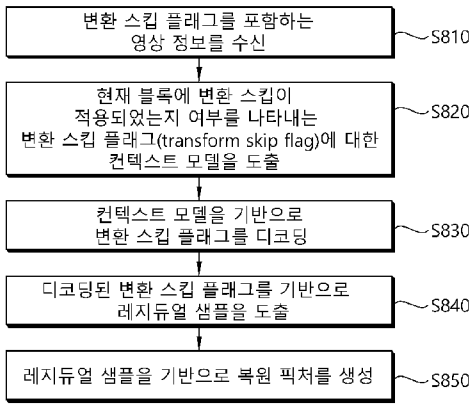
(74) 대리인: 인비전 특허법인 (**ENVISION PATENT & LAW FIRM**); 06193 서울시 강남구 테헤란로 70길 16, 8층, Seoul (KR).

(81) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 국내 권리의 보호를 위하여): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 역내 권리의 보호를 위하여): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 유라시아 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 유럽 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

(54) **Title:** IMAGE DECODING METHOD AND DEVICE USING TRANSFORM SKIP FLAG IN IMAGE CODING SYSTEM

(54) **발명의 명칭:** 영상 코딩 시스템에서 변환 스킵 플래그를 이용한 영상 디코딩 방법 및 그 장치



S810 ... Receive image information including transform skip flag
S820 ... Derive context model for transform skip flag indicating whether transform skip is applied to current block
S830 ... Decode transform skip flag on basis of context model
S840 ... Derive residual sample on basis of decoded transform skip flag
S850 ... Generate reconstruction picture on basis of residual sample

(57) Abstract: An image decoding method performed by a decoding device according the present document comprises the steps of: receiving image information including a transform skip flag; deriving a context model for the transform skip flag indicating whether a transform skip is applied to a current block; decoding the transform skip flag on the basis of the context model; deriving a residual sample on the basis of the decoded transform skip flag; and generating a reconstruction picture on the basis of the residual sample.

(57) 요약서: 본 문서에 따른 디코딩 장치에 의하여 수행되는 영상 디코딩 방법은 변환 스킵 플래그를 포함하는 영상 정보를 수신하는 단계, 현재 블록에 변환 스킵이 적용되었는지 여부를 나타내는 변환 스킵 플래그(transform skip flag)에 대한 컨텍스트 모델을 도출하는 단계, 컨텍스트 모델을 기반으로 변환 스킵 플래그를 디코딩하는 단계, 디코딩된 변환 스킵 플래그를 기반으로 레지듀얼 샘플을 도출하는 단계 및 레지듀얼 샘플을 기반으로 복원 픽처를 생성하는 단계를 포함한다.

WO 2020/189971 A1

[다음 쪽 계속]

공개:
— 국제조사보고서와 함께 (조약 제21조(3))

명세서

발명의 명칭: 영상 코딩 시스템에서 변환 스킵 플래그를 이용한 영상 디코딩 방법 및 그 장치

기술분야

[1] 본 문서는 영상 코딩 기술에 관한 것으로서 영상 코딩 시스템에서 변환 스킵 플래그의 컨텍스트 모델을 도출하고, 도출된 컨텍스트 모델을 기반으로 상기 변환 스킵 플래그를 코딩하는 영상 디코딩 방법 및 그 장치에 관한 것이다.

배경기술

[2] 최근 HD(High Definition) 영상 및 UHD(Ultra High Definition) 영상과 같은 고해상도, 고품질의 영상에 대한 수요가 다양한 분야에서 증가하고 있다. 영상 데이터가 고해상도, 고품질이 될수록 기존의 영상 데이터에 비해 상대적으로 전송되는 정보량 또는 비트량이 증가하기 때문에 기존의 유무선 광대역 회선과 같은 매체를 이용하여 영상 데이터를 전송하거나 기존의 저장 매체를 이용해 영상 데이터를 저장하는 경우, 전송 비용과 저장 비용이 증가된다.

[3] 이에 따라, 고해상도, 고품질 영상의 정보를 효과적으로 전송하거나 저장하고, 재생하기 위해 고효율의 영상 압축 기술이 요구된다.

발명의 상세한 설명

기술적 과제

[4] 본 문서의 기술적 과제는 영상 코딩 효율을 높이는 방법 및 장치를 제공함에 있다.

[5] 본 문서의 다른 기술적 과제는 레지듀얼 코딩의 효율을 높이는 방법 및 장치를 제공함에 있다.

[6] 본 문서의 또 다른 기술적 과제는 레지듀얼 정보을 코딩함에 있어서 변환 스킵 플래그의 컨텍스트 모델을 현재 블록의 예측 모드 정보 또는 상기 현재 블록의 주변 블록들 중 적어도 하나를 기반으로 도출하여 코딩하는 방법 및 장치를 제공함에 있다.

과제 해결 수단

[7] 본 문서의 일 실시예에 따르면, 디코딩 장치에 의하여 수행되는 영상 디코딩 방법이 제공된다. 상기 방법은 변환 스킵 플래그를 포함하는 영상 정보를 수신하는 단계; 현재 블록에 변환 스킵이 적용되었는지 여부를 나타내는 상기 변환 스킵 플래그(transform skip flag)에 대한 컨텍스트 모델을 도출하는 단계;

[8] 상기 컨텍스트 모델을 기반으로 상기 변환 스킵 플래그를 디코딩하는 단계; 상기 디코딩된 변환 스킵 플래그를 기반으로 레지듀얼 샘플을 도출하는 단계; 및 상기 레지듀얼 샘플을 기반으로 복원 픽처를 생성하는 단계를 포함하고, 상기 변환 스킵 플래그에 대한 상기 컨텍스트 모델은 상기 변환 스킵 플래그에 대한 컨텍스트 인덱스 증분(context index increment)을 기반으로 결정되고, 상기 변환

스킵 플래그에 대한 상기 컨텍스트 인덱스 증분은, 상기 현재 블록의 예측 모드 정보 또는 상기 현재 블록의 주변 블록들 중 적어도 하나에 기초하여 도출 된다.

[9] 본 문서의 다른 일 실시예에 따르면, 인코딩 장치에 의하여 수행되는 비디오 인코딩 방법을 제공한다. 상기 방법은 현재 블록에 변환 스킵이 적용되었는지 여부를 나타내는 변환 스킵 플래그(transform skip flag)에 대한 컨텍스트 모델을 도출하는 단계; 상기 컨텍스트 모델을 기반으로 상기 변환 스킵 플래그를 인코딩하는 단계; 및 상기 인코딩된 변환 스킵 플래그를 포함하는 인코딩된 영상 정보를 출력하는 단계를 포함하고, 상기 변환 스킵 플래그에 대한 상기 컨텍스트 모델은 상기 변환 스킵 플래그에 대한 컨텍스트 인덱스 증분(context index increment)을 기반으로 결정되고, 상기 변환 스킵 플래그에 대한 상기 컨텍스트 인덱스 증분은, 상기 현재 블록의 예측 모드 정보 또는 상기 현재 블록의 주변 블록들 중 적어도 하나에 기초하여 도출된다.

[10] 본 문서의 또 다른 일 실시예에 따르면, 디코딩 장치로 하여금 영상 디코딩 방법을 수행하도록 야기하는 영상 정보를 포함하는 비트스트림이 저장된 컴퓨터 판독가능 디지털 저장 매체를 제공한다. 상기 영상 디코딩 방법은, 변환 스킵 플래그를 포함하는 영상 정보를 수신하는 단계; 현재 블록에 변환 스킵이 적용되었는지 여부를 나타내는 상기 변환 스킵 플래그(transform skip flag)에 대한 컨텍스트 모델을 도출하는 단계; 상기 컨텍스트 모델을 기반으로 상기 변환 스킵 플래그를 디코딩하는 단계; 상기 디코딩된 변환 스킵 플래그를 기반으로 레지듀얼 샘플을 도출하는 단계; 및 상기 레지듀얼 샘플을 기반으로 복원 픽처를 생성하는 단계를 포함하고, 상기 변환 스킵 플래그에 대한 상기 컨텍스트 모델은 상기 변환 스킵 플래그에 대한 컨텍스트 인덱스 증분(context index increment)을 기반으로 결정되고, 상기 변환 스킵 플래그에 대한 상기 컨텍스트 인덱스 증분은, 상기 현재 블록의 예측 모드 정보 또는 상기 현재 블록의 주변 블록들 중 적어도 하나에 기초하여 도출된다.

발명의 효과

[11] 본 문서에 따르면 전반적인 영상/비디오 압축 효율을 높일 수 있다.

[12] 본 문서에 따르면 레지듀얼 코딩의 효율을 높일 수 있다.

[13] 본 문서에 따르면 변환 스킵 플래그를 컨텍스트 모델을 기반으로 코딩하고, 이를 통하여 변환 스킵 플래그에 할당되는 비트량을 절약하고, 전반적인 레지듀얼 코딩 효율을 향상시킬 수 있다.

[14] 본 문서에 따르면 변환 스킵 플래그의 컨텍스트 모델을 현재 블록의 예측 모드 정보 또는 현재 블록의 주변 블록들 중 적어도 하나를 기반으로 도출하여 상기 변환 스킵 플래그에 할당되는 비트량을 절약하고, 전반적인 레지듀얼 코딩 효율을 향상시킬 수 있다.

도면의 간단한 설명

[15] 도 1은 본 문서의 실시예들이 적용될 수 있는 비디오/영상 코딩 시스템의 예를

개략적으로 나타낸다.

[16] 도 2는 본 문서의 실시예들이 적용될 수 있는 비디오/영상 인코딩 장치의 구성을 개략적으로 설명하는 도면이다.

[17] 도 3은 본 문서의 실시예들이 적용될 수 있는 비디오/영상 디코딩 장치의 구성을 개략적으로 설명하는 도면이다.

[18] 도 4는 신텍스 엘리먼트(syntax element)를 인코딩하기 위한 CABAC(context-adaptive binary arithmetic coding)을 예시적으로 나타낸다.

[19] 도 5는 4x4 블록 내 변환 계수들의 예시를 도시하는 도면이다.

[20] 도 6은 본 문서에 따른 인코딩 장치에 의한 영상 인코딩 방법을 개략적으로 나타낸다.

[21] 도 7은 본 문서에 따른 영상 인코딩 방법을 수행하는 인코딩 장치를 개략적으로 나타낸다.

[22] 도 8은 본 문서에 따른 디코딩 장치에 의한 영상 디코딩 방법을 개략적으로 나타낸다.

[23] 도 9는 본 문서에 따른 영상 디코딩 방법을 수행하는 디코딩 장치를 개략적으로 나타낸다.

[24] 도 10은 본 문서의 실시예들이 적용되는 컨텐츠 스트리밍 시스템 구조도를 예시적으로 나타낸다.

발명의 실시를 위한 최선의 형태

[25] 본 개시는 다양한 변경을 가할 수 있고 여러 가지 실시예를 가질 수 있는 바, 특정 실시예들을 도면에 예시하고 상세하게 설명하고자 한다. 그러나, 이는 본 개시를 특정 실시예에 한정하려고 하는 것이 아니다. 본 명세서에서 상용하는 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로, 본 개시의 기술적 사상을 한정하려는 의도로 사용되는 것은 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서 "포함하다" 또는 "가지다" 등의 용어는 명세서상에 기재된 특징, 숫자, 단계, 동작, 구성 요소, 부품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성 요소, 부품 도는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[26] 한편, 본 개시에서 설명되는 도면상의 각 구성들은 서로 다른 특징적인 기능들에 관한 설명의 편의를 위해 독립적으로 도시된 것으로서, 각 구성들이 서로 별개의 하드웨어나 별개의 소프트웨어로 구현된다는 것을 의미하지는 않는다. 예컨대, 각 구성 중 두 개 이상의 구성이 합쳐져 하나의 구성을 이룰 수도 있고, 하나의 구성이 복수의 구성으로 나뉘어질 수도 있다. 각 구성이 통합 및/또는 분리된 실시예도 본 개시의 본질에서 벗어나지 않는 한 본 개시의 권리범위에 포함된다.

[27] 본 명세서에서 “A 또는 B(A or B)”는 “오직 A”, “오직 B” 또는 “A와 B 모두”를 의미할 수 있다. 달리 표현하면, 본 명세서에서 “A 또는 B(A or B)”는 “A 및/또는 B(A and/or B)”으로 해석될 수 있다. 예를 들어, 본 명세서에서 “A, B 또는 C(A, B or C)”는 “오직 A”, “오직 B”, “오직 C”, 또는 “A, B 및 C의 임의의 모든 조합(any combination of A, B and C)”를 의미할 수 있다.

[28] 본 명세서에서 사용되는 슬래쉬(/)나 쉼표(comma)는 “및/또는(and/or)”을 의미할 수 있다. 예를 들어, “A/B”는 “A 및/또는 B”를 의미할 수 있다. 이에 따라 “A/B”는 “오직 A”, “오직 B”, 또는 “A와 B 모두”를 의미할 수 있다. 예를 들어, “A, B, C”는 “A, B 또는 C”를 의미할 수 있다.

[29] 본 명세서에서 “적어도 하나의 A 및 B(at least one of A and B)”는, “오직 A”, “오직 B” 또는 “A와 B 모두”를 의미할 수 있다. 또한, 본 명세서에서 “적어도 하나의 A 또는 B(at least one of A or B)”나 “적어도 하나의 A 및/또는 B(at least one of A and/or B)”라는 표현은 “적어도 하나의 A 및 B(at least one of A and B)”와 동일하게 해석될 수 있다.

[30] 또한, 본 명세서에서 “적어도 하나의 A, B 및 C(at least one of A, B and C)”는, “오직 A”, “오직 B”, “오직 C”, 또는 “A, B 및 C의 임의의 모든 조합(any combination of A, B and C)”를 의미할 수 있다. 또한, “적어도 하나의 A, B 또는 C(at least one of A, B or C)”나 “적어도 하나의 A, B 및/또는 C(at least one of A, B and/or C)”는 “적어도 하나의 A, B 및 C(at least one of A, B and C)”를 의미할 수 있다.

[31] 또한, 본 명세서에서 사용되는 괄호는 “예를 들어(for example)”를 의미할 수 있다. 구체적으로, “예측(인트라 예측)”로 표시된 경우, “예측”의 일례로 “인트라 예측”이 제안된 것일 수 있다. 달리 표현하면 본 명세서의 “예측”은 “인트라 예측”으로 제한(limit)되지 않고, “인트라 예측”이 “예측”의 일례로 제안될 것일 수 있다. 또한, “예측(즉, 인트라 예측)”으로 표시된 경우에도, “예측”의 일례로 “인트라 예측”이 제안된 것일 수 있다.

[32] 본 명세서에서 하나의 도면 내에서 개별적으로 설명되는 기술적 특징은, 개별적으로 구현될 수도 있고, 동시에 구현될 수도 있다.

[33] 이하, 첨부한 도면들을 참조하여, 본 개시의 바람직한 실시예를 보다 상세하게 설명하고자 한다. 이하, 도면상의 동일한 구성 요소에 대해서는 동일한 참조 부호를 사용하고 동일한 구성 요소에 대해서 중복된 설명은 생략될 수 있다.

[34] 도 1은 본 개시가 적용될 수 있는 비디오/영상 코딩 시스템의 예를 개략적으로 나타낸다.

[35] 도 1을 참조하면, 비디오/영상 코딩 시스템은 제1 장치(소스 디바이스) 및 제2 장치(수신 디바이스)를 포함할 수 있다. 소스 디바이스는 인코딩된 비디오(video)/영상(image) 정보 또는 데이터를 파일 또는 스트리밍 형태로 디지털 저장매체 또는 네트워크를 통하여 수신 디바이스로 전달할 수 있다.

[36] 상기 소스 디바이스는 비디오 소스, 인코딩 장치, 전송부를 포함할 수 있다.

상기 수신 디바이스는 수신부, 디코딩 장치 및 렌더러를 포함할 수 있다. 상기 인코딩 장치는 비디오/영상 인코딩 장치라고 불릴 수 있고, 상기 디코딩 장치는 비디오/영상 디코딩 장치라고 불릴 수 있다. 송신기는 인코딩 장치에 포함될 수 있다. 수신기는 디코딩 장치에 포함될 수 있다. 렌더러는 디스플레이부를 포함할 수도 있고, 디스플레이부는 별개의 디바이스 또는 외부 컴포넌트로 구성될 수도 있다.

[37] 비디오 소스는 비디오/영상의 캡쳐, 합성 또는 생성 과정 등을 통하여 비디오/영상을 획득할 수 있다. 비디오 소스는 비디오/영상 캡쳐 디바이스 및/또는 비디오/영상 생성 디바이스를 포함할 수 있다. 비디오/영상 캡쳐 디바이스는 예를 들어, 하나 이상의 카메라, 이전에 캡쳐된 비디오/영상을 포함하는 비디오/영상 아카이브 등을 포함할 수 있다. 비디오/영상 생성 디바이스는 예를 들어 컴퓨터, 타블렛 및 스마트폰 등을 포함할 수 있으며 (전자적으로) 비디오/영상을 생성할 수 있다. 예를 들어, 컴퓨터 등을 통하여 가상의 비디오/영상이 생성될 수 있으며, 이 경우 관련 데이터가 생성되는 과정으로 비디오/영상 캡쳐 과정이 갈음될 수 있다.

[38] 인코딩 장치는 입력 비디오/영상을 인코딩할 수 있다. 인코딩 장치는 압축 및 코딩 효율을 위하여 예측, 변환, 양자화 등 일련의 절차를 수행할 수 있다. 인코딩된 데이터(인코딩된 비디오/영상 정보)는 비트스트림(bitstream) 형태로 출력될 수 있다.

[39] 전송부는 비트스트림 형태로 출력된 인코딩된 비디오/영상 정보 또는 데이터를 파일 또는 스트리밍 형태로 디지털 저장매체 또는 네트워크를 통하여 수신 디바이스의 수신부로 전달할 수 있다. 디지털 저장 매체는 USB, SD, CD, DVD, 블루레이, HDD, SSD 등 다양한 저장 매체를 포함할 수 있다. 전송부는 미리 정해진 파일 포멧을 통하여 미디어 파일을 생성하기 위한 엘리먼트를 포함할 수 있고, 방송/통신 네트워크를 통한 전송을 위한 엘리먼트를 포함할 수 있다. 수신부는 상기 비트스트림을 수신/추출하여 디코딩 장치로 전달할 수 있다.

[40] 디코딩 장치는 인코딩 장치의 동작에 대응하는 역양자화, 역변환, 예측 등 일련의 절차를 수행하여 비디오/영상을 디코딩할 수 있다.

[41] 렌더러는 디코딩된 비디오/영상을 렌더링할 수 있다. 렌더링된 비디오/영상은 디스플레이부를 통하여 디스플레이될 수 있다.

[42] 이 문서는 비디오/영상 코딩에 관한 것이다. 예를 들어 이 문서에서 개시된 방법/실시예는 VVC (versatile video coding) 표준, EVC (essential video coding) 표준, AV1 (AOMedia Video 1) 표준, AVS2 (2nd generation of audio video coding standard) 또는 차세대 비디오/영상 코딩 표준(ex. H.267 or H.268 등)에 개시되는 방법에 적용될 수 있다.

[43] 이 문서에서는 비디오/영상 코딩에 관한 다양한 실시예들을 제시하며, 다른 언급이 없는 한 상기 실시예들은 서로 조합되어 수행될 수도 있다.

[44] 이 문서에서 비디오(video)는 시간의 흐름에 따른 일련의 영상(image)들의 집합을 의미할 수 있다. 픽처(picture)는 일반적으로 특정 시간대의 하나의 영상을 나타내는 단위를 의미하며, 슬라이스(slice)/타일(tile)는 코딩에 있어서 픽처의 일부를 구성하는 단위이다. 슬라이스/타일은 하나 이상의 CTU(coding tree unit)을 포함할 수 있다. 하나의 픽처는 하나 이상의 슬라이스/타일로 구성될 수 있다.

[45] 타일은 특정 타일 열 및 특정 타일 열 이내의 CTU들의 사각 영역이다(A tile is a rectangular region of CTUs within a particular tile column and a particular tile row in a picture). 상기 타일 열은 CTU들의 사각 영역이고, 상기 사각 영역은 상기 픽처의 높이와 동일한 높이를 갖고, 너비는 픽처 파라미터 세트 내의 신택스 요소들에 의하여 명시될 수 있다(The tile column is a rectangular region of CTUs having a height equal to the height of the picture and a width specified by syntax elements in the picture parameter set). 상기 타일 행은 CTU들의 사각 영역이고, 상기 사각 영역은 픽처 파라미터 세트 내의 신택스 요소들에 의하여 명시되는 너비를 갖고, 높이는 상기 픽처의 높이와 동일할 수 있다(The tile row is a rectangular region of CTUs having a height specified by syntax elements in the picture parameter set and a width equal to the width of the picture). 타일 스캔은 픽처를 파티셔닝하는 CTU들의 특정 순차적 오더링을 나타낼 수 있고, 상기 CTU들은 타일 내 CTU 래스터 스캔으로 연속적으로 정렬될 수 있고, 픽처 내 타일들은 상기 픽처의 상기 타일들의 래스터 스캔으로 연속적으로 정렬될 수 있다(A tile scan is a specific sequential ordering of CTUs partitioning a picture in which the CTUs are ordered consecutively in CTU raster scan in a tile whereas tiles in a picture are ordered consecutively in a raster scan of the tiles of the picture). 슬라이스는 다수의 완전한 타일들 또는 하나의 NAL 유닛에 포함될 수 있는 픽처의 하나의 타일 내 다수의 연속적인 CTU 행들을 포함할 수 있다. 이 문서에서 타일 그룹과 슬라이스는 혼용될 수 있다. 예를 들어 본 문서에서 tile group/tile group header는 slice/slice header로 불리 수 있다.

[46] 한편, 하나의 픽처는 둘 이상의 서브픽처로 구분될 수 있다. 서브픽처는 픽처 내 하나 이상의 슬라이스들의 사각 리전일 수 있다(an rectangular region of one or more slices within a picture).

[47] 픽셀(pixel) 또는 펠(pel)은 하나의 픽처(또는 영상)을 구성하는 최소의 단위를 의미할 수 있다. 또한, 픽셀에 대응하는 용어로서 '샘플(sample)'이 사용될 수 있다. 샘플은 일반적으로 픽셀 또는 픽셀의 값을 나타낼 수 있으며, 루마(luma) 성분의 픽셀/픽셀값만을 나타낼 수도 있고, 크로마(chroma) 성분의 픽셀/픽셀 값만을 나타낼 수도 있다.

[48] 유닛(unit)은 영상 처리의 기본 단위를 나타낼 수 있다. 유닛은 픽처의 특정 영역 및 해당 영역에 관련된 정보 중 적어도 하나를 포함할 수 있다. 하나의 유닛은 하나의 루마 블록 및 두개의 크로마(ex. cb, cr) 블록을 포함할 수 있다. 유닛은 경우에 따라서 블록(block) 또는 영역(area) 등의 용어와 혼용하여 사용될 수 있다.

일반적인 경우, MxN 블록은 M개의 열과 N개의 행으로 이루어진 샘플들(또는 샘플 어레이) 또는 변환 계수(transform coefficient)들의 집합(또는 어레이)을 포함할 수 있다.

[49] 도 2는 본 개시가 적용될 수 있는 비디오/영상 인코딩 장치의 구성을 개략적으로 설명하는 도면이다. 이하 비디오 인코딩 장치라 함은 영상 인코딩 장치를 포함할 수 있다.

[50] 도 2를 참조하면, 인코딩 장치(200)는 영상 분할부(image partitioner, 210), 예측부(predictor, 220), 레지듀얼 처리부(residual processor, 230), 엔트로피 인코딩부(entropy encoder, 240), 가산부(adder, 250), 필터링부(filter, 260) 및 메모리(memory, 270)를 포함하여 구성될 수 있다. 예측부(220)는 인터 예측부(221) 및 인트라 예측부(222)를 포함할 수 있다. 레지듀얼 처리부(230)는 변환부(transformer, 232), 양자화부(quantizer 233), 역양자화부(dequantizer 234), 역변환부(inverse transformer, 235)를 포함할 수 있다. 레지듀얼 처리부(230)은 감산부(subtractor, 231)를 더 포함할 수 있다. 가산부(250)는 복원부(reconstructor) 또는 복원 블록 생성부(recontructged block generator)로 불릴 수 있다. 상술한 영상 분할부(210), 예측부(220), 레지듀얼 처리부(230), 엔트로피 인코딩부(240), 가산부(250) 및 필터링부(260)는 실시예에 따라 하나 이상의 하드웨어 컴포넌트(예를 들어 인코더 칩셋 또는 프로세서)에 의하여 구성될 수 있다. 또한 메모리(270)는 DPB(decoded picture buffer)를 포함할 수 있고, 디지털 저장 매체에 의하여 구성될 수도 있다. 상기 하드웨어 컴포넌트는 메모리(270)을 내/외부 컴포넌트로 더 포함할 수도 있다.

[51] 영상 분할부(210)는 인코딩 장치(200)에 입력된 입력 영상(또는, 픽쳐, 프레임)를 하나 이상의 처리 유닛(processing unit)으로 분할할 수 있다. 일 예로, 상기 처리 유닛은 코딩 유닛(coding unit, CU)이라고 불릴 수 있다. 이 경우 코딩 유닛은 코딩 트리 유닛(coding tree unit, CTU) 또는 최대 코딩 유닛(largest coding unit, LCU)으로부터 QTBTTT (Quad-tree binary-tree ternary-tree) 구조에 따라 재귀적으로(recursively) 분할될 수 있다. 예를 들어, 하나의 코딩 유닛은 쿼드 트리 구조, 바이너리 트리 구조, 및/또는 터너리 구조를 기반으로 하위(deeper) 뎁스의 복수의 코딩 유닛들로 분할될 수 있다. 이 경우 예를 들어 쿼드 트리 구조가 먼저 적용되고 바이너리 트리 구조 및/또는 터너리 구조가 나중에 적용될 수 있다. 또는 바이너리 트리 구조가 먼저 적용될 수도 있다. 더 이상 분할되지 않는 최종 코딩 유닛을 기반으로 본 개시에 따른 코딩 절차가 수행될 수 있다. 이 경우 영상 특성에 따른 코딩 효율 등을 기반으로, 최대 코딩 유닛이 바로 최종 코딩 유닛으로 사용될 수 있고, 또는 필요에 따라 코딩 유닛은 재귀적으로(recursively) 보다 하위 뎁스의 코딩 유닛들로 분할되어 최적의 사이즈의 코딩 유닛이 최종 코딩 유닛으로 사용될 수 있다. 여기서 코딩 절차라 함은 후술하는 예측, 변환, 및 복원 등의 절차를 포함할 수 있다. 다른 예로, 상기 처리 유닛은 예측 유닛(PU: Prediction Unit) 또는 변환 유닛(TU: Transform Unit)을

더 포함할 수 있다. 이 경우 상기 예측 유닛 및 상기 변환 유닛은 각각 상술한 최종 코딩 유닛으로부터 분할 또는 파티셔닝될 수 있다. 상기 예측 유닛은 샘플 예측의 단위일 수 있고, 상기 변환 유닛은 변환 계수를 유도하는 단위 및/또는 변환 계수로부터 레지듀얼 신호(residual signal)를 유도하는 단위일 수 있다.

[52] 유닛은 경우에 따라서 블록(block) 또는 영역(area) 등의 용어와 혼용하여 사용될 수 있다. 일반적인 경우, MxN 블록은 M개의 열과 N개의 행으로 이루어진 샘플들 또는 변환 계수(transform coefficient)들의 집합을 나타낼 수 있다. 샘플은 일반적으로 픽셀 또는 픽셀의 값을 나타낼 수 있으며, 휘도(luma) 성분의 픽셀/픽셀값만을 나타낼 수도 있고, 채도(chroma) 성분의 픽셀/픽셀 값만을 나타낼 수도 있다. 샘플은 하나의 픽처(또는 영상)을 픽셀(pixel) 또는 펠(pel)에 대응하는 용어로서 사용될 수 있다.

[53] 인코딩 장치(200)는 입력 영상 신호(원본 블록, 원본 샘플 어레이)에서 인터 예측부(221) 또는 인트라 예측부(222)로부터 출력된 예측 신호(예측된 블록, 예측 샘플 어레이)를 감산하여 레지듀얼 신호(residual signal, 잔여 블록, 잔여 샘플 어레이)를 생성할 수 있고, 생성된 레지듀얼 신호는 변환부(232)로 전송된다. 이 경우 도시된 바와 같이 인코딩 장치(200) 내에서 입력 영상 신호(원본 블록, 원본 샘플 어레이)에서 예측 신호(예측 블록, 예측 샘플 어레이)를 감산하는 유닛은 감산부(231)라고 불릴 수 있다. 예측부는 처리 대상 블록(이하, 현재 블록이라 함)에 대한 예측을 수행하고, 상기 현재 블록에 대한 예측 샘플들을 포함하는 예측된 블록(predicted block)을 생성할 수 있다. 예측부는 현재 블록 또는 CU 단위로 인트라 예측이 적용되는지 또는 인터 예측이 적용되는지 결정할 수 있다. 예측부는 각 예측모드에 대한 설명에서 후술하는 바와 같이 예측 모드 정보 등 예측에 관한 다양한 정보를 생성하여 엔트로피 인코딩부(240)로 전달할 수 있다. 예측에 관한 정보는 엔트로피 인코딩부(240)에서 인코딩되어 비트스트림 형태로 출력될 수 있다.

[54] 인트라 예측부(222)는 현재 픽처 내의 샘플들을 참조하여 현재 블록을 예측할 수 있다. 상기 참조되는 샘플들은 예측 모드에 따라 상기 현재 블록의 주변(neighbor)에 위치할 수 있고, 또는 떨어져서 위치할 수도 있다. 인트라 예측에서 예측 모드들은 복수의 비방향성 모드와 복수의 방향성 모드를 포함할 수 있다. 비방향성 모드는 예를 들어 DC 모드 및 플래너 모드(Planar 모드)를 포함할 수 있다. 방향성 모드는 예측 방향의 세밀한 정도에 따라 예를 들어 33개의 방향성 예측 모드 또는 65개의 방향성 예측 모드를 포함할 수 있다. 다만, 이는 예시로서 설정에 따라 그 이상 또는 그 이하의 개수의 방향성 예측 모드들이 사용될 수 있다. 인트라 예측부(222)는 주변 블록에 적용된 예측 모드를 이용하여, 현재 블록에 적용되는 예측 모드를 결정할 수도 있다.

[55] 인터 예측부(221)는 참조 픽처 상에서 움직임 벡터에 의해 특정되는 참조 블록(참조 샘플 어레이)을 기반으로, 현재 블록에 대한 예측된 블록을 유도할 수 있다. 이때, 인터 예측 모드에서 전송되는 움직임 정보의 양을 줄이기 위해 주변

블록과 현재 블록 간의 움직임 정보의 상관성에 기초하여 움직임 정보를 블록, 서브블록 또는 샘플 단위로 예측할 수 있다. 상기 움직임 정보는 움직임 벡터 및 참조 픽처 인덱스를 포함할 수 있다. 상기 움직임 정보는 인터 예측 방향(L0 예측, L1 예측, Bi 예측 등) 정보를 더 포함할 수 있다. 인터 예측의 경우에, 주변 블록은 현재 픽처 내에 존재하는 공간적 주변 블록(spatial neighboring block)과 참조 픽처에 존재하는 시간적 주변 블록(temporal neighboring C)을 포함할 수 있다. 상기 참조 블록을 포함하는 참조 픽처와 상기 시간적 주변 블록을 포함하는 참조 픽처는 동일할 수도 있고, 다를 수도 있다. 상기 시간적 주변 블록은 동일 위치 참조 블록(collocated reference block), 동일 위치 CU(colCU) 등의 이름으로 불릴 수 있으며, 상기 시간적 주변 블록을 포함하는 참조 픽처는 동일 위치 픽처(collocated picture, colPic)라고 불릴 수도 있다. 예를 들어, 인터 예측부(221)는 주변 블록들을 기반으로 움직임 정보 후보 리스트를 구성하고, 상기 현재 블록의 움직임 벡터 및/또는 참조 픽처 인덱스를 도출하기 위하여 어떤 후보가 사용되는지를 지시하는 정보를 생성할 수 있다. 다양한 예측 모드를 기반으로 인터 예측이 수행될 수 있으며, 예를 들어 스킵 모드와 머지 모드의 경우에, 인터 예측부(221)는 주변 블록의 움직임 정보를 현재 블록의 움직임 정보로 이용할 수 있다. 스킵 모드의 경우, 머지 모드와 달리 레지듀얼 신호가 전송되지 않을 수 있다. 움직임 정보 예측(motion vector prediction, MVP) 모드의 경우, 주변 블록의 움직임 벡터를 움직임 벡터 예측자(motion vector predictor)로 이용하고, 움직임 벡터 차분(motion vector difference)을 시그널링함으로써 현재 블록의 움직임 벡터를 지시할 수 있다.

[56] 예측부(220)는 후술하는 다양한 예측 방법을 기반으로 예측 신호를 생성할 수 있다. 예를 들어, 예측부는 하나의 블록에 대한 예측을 위하여 인트라 예측 또는 인터 예측을 적용할 수 있을 뿐 아니라, 인트라 예측과 인터 예측을 동시에 적용할 수 있다. 이는 combined inter and intra prediction (CIIP)라고 불릴 수 있다. 또한, 예측부는 블록에 대한 예측을 위하여 인트라 블록 카피(intra block copy, IBC) 예측 모드에 기반할 수도 있고 또는 팔레트 모드(palette mode)에 기반할 수도 있다. 상기 IBC 예측 모드 또는 팔레트 모드는 예를 들어 SCC(screen content coding) 등과 같이 게임 등의 컨텐츠 영상/동영상 코딩을 위하여 사용될 수 있다. IBC는 기본적으로 현재 픽처 내에서 예측을 수행하나 현재 픽처 내에서 참조 블록을 도출하는 점에서 인터 예측과 유사하게 수행될 수 있다. 즉, IBC는 본 문서에서 설명되는 인터 예측 기법들 중 적어도 하나를 이용할 수 있다. 팔레트 모드는 인트라 코딩 또는 인트라 예측의 일 예로 볼 수 있다. 팔레트 모드가 적용되는 경우 팔레트 테이블 및 팔레트 인덱스에 관한 정보를 기반으로 픽처 내 샘플 값을 시그널링할 수 있다.

[57] 상기 예측부 (인터 예측부(221) 및/또는 상기 인트라 예측부(222) 포함)를 통해 생성된 예측 신호는 복원 신호를 생성하기 위해 이용되거나 레지듀얼 신호를 생성하기 위해 이용될 수 있다. 변환부(232)는 레지듀얼 신호에 변환 기법을

적용하여 변환 계수들(transform coefficients)를 생성할 수 있다. 예를 들어, 변환 기법은 DCT(Discrete Cosine Transform), DST(Discrete Sine Transform), KLT(Karhunen-Loeve Transform), GBT(Graph-Based Transform), 또는 CNT(Conditionally Non-linear Transform) 중 적어도 하나를 포함할 수 있다. 여기서, GBT는 픽셀 간의 관계 정보를 그래프로 표현한다고 할 때 이 그래프로부터 얻어진 변환을 의미한다. CNT는 이전에 복원된 모든 픽셀(all previously reconstructed pixel)를 이용하여 예측 신호를 생성하고 그에 기초하여 획득되는 변환을 의미한다. 또한, 변환 과정은 정사각형의 동일한 크기를 갖는 픽셀 블록에 적용될 수도 있고, 정사각형이 아닌 가변 크기의 블록에도 적용될 수 있다.

[58] 양자화부(233)는 변환 계수들을 양자화하여 엔트로피 인코딩부(240)로 전송되고, 엔트로피 인코딩부(240)는 양자화된 신호(양자화된 변환 계수들에 관한 정보)를 인코딩하여 비트스트림으로 출력할 수 있다. 상기 양자화된 변환 계수들에 관한 정보는 레지듀얼 정보라고 불릴 수 있다. 양자화부(233)는 계수 스캔 순서(scan order)를 기반으로 블록 형태의 양자화된 변환 계수들을 1차원 벡터 형태로 재정렬할 수 있고, 상기 1차원 벡터 형태의 양자화된 변환 계수들을 기반으로 상기 양자화된 변환 계수들에 관한 정보를 생성할 수도 있다. 엔트로피 인코딩부(240)는 예를 들어 지수 골롬(exponential Golomb), CAVLC(context-adaptive variable length coding), CABAC(context-adaptive binary arithmetic coding) 등과 같은 다양한 인코딩 방법을 수행할 수 있다. 엔트로피 인코딩부(240)는 양자화된 변환 계수들 외 비디오/이미지 복원에 필요한 정보들(예컨대 신택스 요소들(syntax elements)의 값 등)을 함께 또는 별도로 인코딩할 수도 있다. 인코딩된 정보(ex. 인코딩된 비디오/영상 정보)는 비트스트림 형태로 NAL(network abstraction layer) 유닛 단위로 전송 또는 저장될 수 있다. 상기 비디오/영상 정보는 어댑테이션 파라미터 세트(APS), 픽처 파라미터 세트(PPS), 시퀀스 파라미터 세트(SPS) 또는 비디오 파라미터 세트(VPS) 등 다양한 파라미터 세트에 관한 정보를 더 포함할 수 있다. 또한 상기 비디오/영상 정보는 일반 제한 정보(general constraint information)을 더 포함할 수 있다. 본 문서에서 인코딩 장치에서 디코딩 장치로 전달/시그널링되는 정보 및/또는 신택스 요소들은 비디오/영상 정보에 포함될 수 있다. 상기 비디오/영상 정보는 상술한 인코딩 절차를 통하여 인코딩되어 상기 비트스트림에 포함될 수 있다. 상기 비트스트림은 네트워크를 통하여 전송될 수 있고, 또는 디지털 저장매체에 저장될 수 있다. 여기서 네트워크는 방송망 및/또는 통신망 등을 포함할 수 있고, 디지털 저장매체는 USB, SD, CD, DVD, 블루레이, HDD, SSD 등 다양한 저장매체를 포함할 수 있다. 엔트로피 인코딩부(240)로부터 출력된 신호는 전송하는 전송부(미도시) 및/또는 저장하는 저장부(미도시)가 인코딩 장치(200)의 내/외부 엘리먼트로서 구성될 수 있고, 또는 전송부는 엔트로피 인코딩부(240)에 포함될 수도 있다.

[59] 양자화부(233)로부터 출력된 양자화된 변환 계수들은 예측 신호를 생성하기 위해 이용될 수 있다. 예를 들어, 양자화된 변환 계수들에 역양자화부(234) 및 역변환부(235)를 통해 역양자화 및 역변환을 적용함으로써 레지듀얼 신호(레지듀얼 블록 or 레지듀얼 샘플들)를 복원할 수 있다. 가산부(155)는 복원된 레지듀얼 신호를 인터 예측부(221) 또는 인트라 예측부(222)로부터 출력된 예측 신호에 더함으로써 복원(reconstructed) 신호(복원 픽처, 복원 블록, 복원 샘플 어레이)가 생성될 수 있다. 스킵 모드가 적용된 경우와 같이 처리 대상 블록에 대한 레지듀얼이 없는 경우, 예측된 블록이 복원 블록으로 사용될 수 있다. 가산부(250)는 복원부 또는 복원 블록 생성부라고 불릴 수 있다. 생성된 복원 신호는 현재 픽처 내 다음 처리 대상 블록의 인트라 예측을 위하여 사용될 수 있고, 후술하는 바와 같이 필터링을 거쳐서 다음 픽처의 인터 예측을 위하여 사용될 수도 있다.

[60] 한편 픽처 인코딩 및/또는 복원 과정에서 LMCS (luma mapping with chroma scaling)가 적용될 수도 있다.

[61] 필터링부(260)는 복원 신호에 필터링을 적용하여 주관적/객관적 화질을 향상시킬 수 있다. 예를 들어 필터링부(260)은 복원 픽처에 다양한 필터링 방법을 적용하여 수정된(modified) 복원 픽처를 생성할 수 있고, 상기 수정된 복원 픽처를 메모리(270), 구체적으로 메모리(270)의 DPB에 저장할 수 있다. 상기 다양한 필터링 방법은 예를 들어, 디블록킹 필터링, 샘플 적응적 오프셋(sample adaptive offset), 적응적 루프 필터(adaptive loop filter), 양방향 필터(bilateral filter) 등을 포함할 수 있다. 필터링부(260)은 각 필터링 방법에 대한 설명에서 후술하는 바와 같이 필터링에 관한 다양한 정보를 생성하여 엔트로피 인코딩부(240)로 전달할 수 있다. 필터링 관한 정보는 엔트로피 인코딩부(240)에서 인코딩되어 비트스트림 형태로 출력될 수 있다.

[62] 메모리(270)에 전송된 수정된 복원 픽처는 인터 예측부(221)에서 참조 픽처로 사용될 수 있다. 인코딩 장치는 이를 통하여 인터 예측이 적용되는 경우, 인코딩 장치(100)와 디코딩 장치에서의 예측 미스매치를 피할 수 있고, 부호화 효율도 향상시킬 수 있다.

[63] 메모리(270) DPB는 수정된 복원 픽처를 인터 예측부(221)에서의 참조 픽처로 사용하기 위해 저장할 수 있다. 메모리(270)는 현재 픽처 내 움직임 정보가 도출된(또는 인코딩된) 블록의 움직임 정보 및/또는 이미 복원된 픽처 내 블록들의 움직임 정보를 저장할 수 있다. 상기 저장된 움직임 정보는 공간적 주변 블록의 움직임 정보 또는 시간적 주변 블록의 움직임 정보로 활용하기 위하여 인터 예측부(221)에 전달할 수 있다. 메모리(270)는 현재 픽처 내 복원된 블록들의 복원 샘플들을 저장할 수 있고, 인트라 예측부(222)에 전달할 수 있다.

[64] 도 3은 본 개시가 적용될 수 있는 비디오/영상 디코딩 장치의 구성을 개략적으로 설명하는 도면이다.

[65] 도 3을 참조하면, 디코딩 장치(300)는 엔트로피 디코딩부(entropy decoder, 310),

레지듀얼 처리부(residual processor, 320), 예측부(predictor, 330), 가산부(adder, 340), 필터링부(filter, 350) 및 메모리(memory, 360)를 포함하여 구성될 수 있다. 예측부(330)는 인트라 예측부(331) 및 인터 예측부(332)를 포함할 수 있다. 레지듀얼 처리부(320)는 역양자화부(dequantizer, 321) 및 역변환부(inverse transformer, 321)를 포함할 수 있다. 상술한 엔트로피 디코딩부(310), 레지듀얼 처리부(320), 예측부(330), 가산부(340) 및 필터링부(350)는 실시예에 따라 하나의 하드웨어 컴포넌트(예를 들어 디코더 칩셋 또는 프로세서)에 의하여 구성될 수 있다. 또한 메모리(360)는 DPB(decoded picture buffer)를 포함할 수 있고, 디지털 저장 매체에 의하여 구성될 수도 있다. 상기 하드웨어 컴포넌트는 메모리(360)을 내/외부 컴포넌트로 더 포함할 수도 있다.

[66] 비디오/영상 정보를 포함하는 비트스트림이 입력되면, 디코딩 장치(300)는 도 3의 인코딩 장치에서 비디오/영상 정보가 처리된 프로세스에 대응하여 영상을 복원할 수 있다. 예를 들어, 디코딩 장치(300)는 상기 비트스트림으로부터 획득한 블록 분할 관련 정보를 기반으로 유닛들/블록들을 도출할 수 있다. 디코딩 장치(300)는 인코딩 장치에서 적용된 처리 유닛을 이용하여 디코딩을 수행할 수 있다. 따라서 디코딩의 처리 유닛은 예를 들어 코딩 유닛일 수 있고, 코딩 유닛은 코딩 트리 유닛 또는 최대 코딩 유닛으로부터 쿼드 트리 구조, 바이너리 트리 구조 및/또는 터너리 트리 구조를 따라서 분할될 수 있다. 코딩 유닛으로부터 하나 이상의 변환 유닛이 도출될 수 있다. 그리고, 디코딩 장치(300)를 통해 디코딩 및 출력된 복원 영상 신호는 재생 장치를 통해 재생될 수 있다.

[67] 디코딩 장치(300)는 도 3의 인코딩 장치로부터 출력된 신호를 비트스트림 형태로 수신할 수 있고, 수신된 신호는 엔트로피 디코딩부(310)를 통해 디코딩될 수 있다. 예를 들어, 엔트로피 디코딩부(310)는 상기 비트스트림을 파싱하여 영상 복원(또는 픽처 복원)에 필요한 정보(ex. 비디오/영상 정보)를 도출할 수 있다. 상기 비디오/영상 정보는 어댑테이션 파라미터 세트(APS), 픽처 파라미터 세트(PPS), 시퀀스 파라미터 세트(SPS) 또는 비디오 파라미터 세트(VPS) 등 다양한 파라미터 세트에 관한 정보를 더 포함할 수 있다. 또한 상기 비디오/영상 정보는 일반 제한 정보(general constraint information)을 더 포함할 수 있다. 디코딩 장치는 상기 파라미터 세트에 관한 정보 및/또는 상기 일반 제한 정보를 더 기반으로 픽처를 디코딩할 수 있다. 본 문서에서 후술되는 시그널링/수신되는 정보 및/또는 신택스 요소들은 상기 디코딩 절차를 통하여 디코딩되어 상기 비트스트림으로부터 획득될 수 있다. 예컨대, 엔트로피 디코딩부(310)는 지수 골롬 부호화, CAVLC 또는 CABAC 등의 코딩 방법을 기초로 비트스트림 내 정보를 디코딩하고, 영상 복원에 필요한 신택스 엘리먼트의 값, 레지듀얼에 관한 변환 계수의 양자화된 값 들을 출력할 수 있다. 보다 상세하게, CABAC 엔트로피 디코딩 방법은, 비트스트림에서 각 신택스 요소에 해당하는 빈을 수신하고, 디코딩 대상 신택스 요소 정보와 주변 및 디코딩 대상 블록의 디코딩 정보 혹은 이전 단계에서 디코딩된 심볼/빈의 정보를 이용하여 문맥(context) 모델을

결정하고, 결정된 문맥 모델에 따라 빈(bin)의 발생 확률을 예측하여 빈의 산술 디코딩(arithmetic decoding)를 수행하여 각 신택스 요소의 값에 해당하는 심볼을 생성할 수 있다. 이때, CABAC 엔트로피 디코딩 방법은 문맥 모델 결정 후 다음 심볼/빈의 문맥 모델을 위해 디코딩된 심볼/빈의 정보를 이용하여 문맥 모델을 업데이트할 수 있다. 엔트로피 디코딩부(310)에서 디코딩된 정보 중 예측에 관한 정보는 예측부(인터 예측부(332) 및 인트라 예측부(331))로 제공되고, 엔트로피 디코딩부(310)에서 엔트로피 디코딩이 수행된 레지듀얼 값, 즉 양자화된 변환 계수들 및 관련 파라미터 정보는 레지듀얼 처리부(320)로 입력될 수 있다. 레지듀얼 처리부(320)는 레지듀얼 신호(레지듀얼 블록, 레지듀얼 샘플들, 레지듀얼 샘플 어레이)를 도출할 수 있다. 또한, 엔트로피 디코딩부(310)에서 디코딩된 정보 중 필터링에 관한 정보는 필터링부(350)으로 제공될 수 있다. 한편, 인코딩 장치로부터 출력된 신호를 수신하는 수신부(미도시)가 디코딩 장치(300)의 내/외부 엘리먼트로서 더 구성될 수 있고, 또는 수신부는 엔트로피 디코딩부(310)의 구성요소일 수도 있다. 한편, 본 문서에 따른 디코딩 장치는 비디오/영상/픽처 디코딩 장치라고 불릴 수 있고, 상기 디코딩 장치는 정보 디코더(비디오/영상/픽처 정보 디코더) 및 샘플 디코더(비디오/영상/픽처 샘플 디코더)로 구분할 수도 있다. 상기 정보 디코더는 상기 엔트로피 디코딩부(310)를 포함할 수 있고, 상기 샘플 디코더는 상기 역양자화부(321), 역변환부(322), 가산부(340), 필터링부(350), 메모리(360), 인터 예측부(332) 및 인트라 예측부(331) 중 적어도 하나를 포함할 수 있다.

[68] 역양자화부(321)에서는 양자화된 변환 계수들을 역양자화하여 변환 계수들을 출력할 수 있다. 역양자화부(321)는 양자화된 변환 계수들을 2차원의 블록 형태로 재정렬할 수 있다. 이 경우 상기 재정렬은 인코딩 장치에서 수행된 계수 스캔 순서를 기반하여 재정렬을 수행할 수 있다. 역양자화부(321)는 양자화 파라미터(예를 들어 양자화 스텝 사이즈 정보)를 이용하여 양자화된 변환 계수들에 대한 역양자화를 수행하고, 변환 계수들(transform coefficient)를 획득할 수 있다.

[69] 역변환부(322)에서는 변환 계수들를 역변환하여 레지듀얼 신호(레지듀얼 블록, 레지듀얼 샘플 어레이)를 획득하게 된다.

[70] 예측부는 현재 블록에 대한 예측을 수행하고, 상기 현재 블록에 대한 예측 샘플들을 포함하는 예측된 블록(predicted block)을 생성할 수 있다. 예측부는 엔트로피 디코딩부(310)로부터 출력된 상기 예측에 관한 정보를 기반으로 상기 현재 블록에 인트라 예측이 적용되는지 또는 인터 예측이 적용되는지 결정할 수 있고, 구체적인 인트라/인터 예측 모드를 결정할 수 있다.

[71] 예측부(330)는 후술하는 다양한 예측 방법을 기반으로 예측 신호를 생성할 수 있다. 예를 들어, 예측부는 하나의 블록에 대한 예측을 위하여 인트라 예측 또는 인터 예측을 적용할 수 있을 뿐 아니라, 인트라 예측과 인터 예측을 동시에 적용할 수 있다. 이는 combined inter and intra prediction (CIIP)라고 불릴 수 있다.

또한, 예측부는 블록에 대한 예측을 위하여 인트라 블록 카피(intra block copy, IBC) 예측 모드에 기반할 수도 있고 또는 팔레트 모드(palette mode)에 기반할 수도 있다. 상기 IBC 예측 모드 또는 팔레트 모드는 예를 들어 SCC(screen content coding) 등과 같이 게임 등의 컨텐츠 영상/동영상 코딩을 위하여 사용될 수 있다. IBC는 기본적으로 현재 픽처 내에서 예측을 수행하나 현재 픽처 내에서 참조 블록을 도출하는 점에서 인터 예측과 유사하게 수행될 수 있다. 즉, IBC는 본 문서에서 설명되는 인터 예측 기법들 중 적어도 하나를 이용할 수 있다. 팔레트 모드는 인트라 코딩 또는 인트라 예측의 일 예로 볼 수 있다. 팔레트 모드가 적용되는 경우 팔레트 테이블 및 팔레트 인덱스에 관한 정보가 상기 비디오/영상 정보에 포함되어 시그널링될 수 있다.

[72] 인트라 예측부(331)는 현재 픽처 내의 샘플들을 참조하여 현재 블록을 예측할 수 있다. 상기 참조되는 샘플들은 예측 모드에 따라 상기 현재 블록의 주변(neighbor)에 위치할 수 있고, 또는 떨어져서 위치할 수도 있다. 인트라 예측에서 예측 모드들은 복수의 비방향성 모드와 복수의 방향성 모드를 포함할 수 있다. 인트라 예측부(331)는 주변 블록에 적용된 예측 모드를 이용하여, 현재 블록에 적용되는 예측 모드를 결정할 수도 있다.

[73] 인터 예측부(332)는 참조 픽처 상에서 움직임 벡터에 의해 특정되는 참조 블록(참조 샘플 어레이)을 기반으로, 현재 블록에 대한 예측된 블록을 유도할 수 있다. 이때, 인터 예측 모드에서 전송되는 움직임 정보의 양을 줄이기 위해 주변 블록과 현재 블록 간의 움직임 정보의 상관성에 기초하여 움직임 정보를 블록, 서브블록 또는 샘플 단위로 예측할 수 있다. 상기 움직임 정보는 움직임 벡터 및 참조 픽처 인덱스를 포함할 수 있다. 상기 움직임 정보는 인터 예측 방향(L0 예측, L1 예측, Bi 예측 등) 정보를 더 포함할 수 있다. 인터 예측의 경우에, 주변 블록은 현재 픽처 내에 존재하는 공간적 주변 블록(spatial neighboring block)과 참조 픽처에 존재하는 시간적 주변 블록(temporal neighboring block)을 포함할 수 있다. 예를 들어, 인터 예측부(332)는 주변 블록들을 기반으로 움직임 정보 후보 리스트를 구성하고, 수신한 후보 선택 정보를 기반으로 상기 현재 블록의 움직임 벡터 및/또는 참조 픽처 인덱스를 도출할 수 있다. 다양한 예측 모드를 기반으로 인터 예측이 수행될 수 있으며, 상기 예측에 관한 정보는 상기 현재 블록에 대한 인터 예측의 모드를 지시하는 정보를 포함할 수 있다.

[74] 가산부(340)는 획득된 레지듀얼 신호를 예측부(인터 예측부(332) 및/또는 인트라 예측부(331) 포함)로부터 출력된 예측 신호(예측된 블록, 예측 샘플 어레이)에 더함으로써 복원 신호(복원 픽처, 복원 블록, 복원 샘플 어레이)를 생성할 수 있다. 스킵 모드가 적용된 경우와 같이 처리 대상 블록에 대한 레지듀얼이 없는 경우, 예측된 블록이 복원 블록으로 사용될 수 있다.

[75] 가산부(340)는 복원부 또는 복원 블록 생성부라고 불릴 수 있다. 생성된 복원 신호는 현재 픽처 내 다음 처리 대상 블록의 인트라 예측을 위하여 사용될 수 있고, 후술하는 바와 같이 필터링을 거쳐서 출력될 수도 있고 또는 다음 픽처의

인터 예측을 위하여 사용될 수도 있다.

[76] 한편, 픽처 디코딩 과정에서 LMCS (luma mapping with chroma scaling)가 적용될 수도 있다.

[77] 필터링부(350)는 복원 신호에 필터링을 적용하여 주관적/객관적 화질을 향상시킬 수 있다. 예를 들어 필터링부(350)는 복원 픽처에 다양한 필터링 방법을 적용하여 수정된(modified) 복원 픽처를 생성할 수 있고, 상기 수정된 복원 픽처를 메모리(360), 구체적으로 메모리(360)의 DPB에 전송할 수 있다. 상기 다양한 필터링 방법은 예를 들어, 디블록킹 필터링, 샘플 적응적 오프셋(sample adaptive offset), 적응적 루프 필터(adaptive loop filter), 양방향 필터(bilateral filter) 등을 포함할 수 있다.

[78] 메모리(360)의 DPB에 저장된 (수정된) 복원 픽처는 인터 예측부(332)에서 참조 픽쳐로 사용될 수 있다. 메모리(360)는 현재 픽처 내 움직임 정보가 도출된(또는 디코딩된) 블록의 움직임 정보 및/또는 이미 복원된 픽처 내 블록들의 움직임 정보를 저장할 수 있다. 상기 저장된 움직임 정보는 공간적 주변 블록의 움직임 정보 또는 시간적 주변 블록의 움직임 정보로 활용하기 위하여 인터 예측부(332)에 전달할 수 있다. 메모리(360)는 현재 픽처 내 복원된 블록들의 복원 샘플들을 저장할 수 있고, 인트라 예측부(331)에 전달할 수 있다.

[79] 본 명세서에서, 인코딩 장치(100)의 필터링부(260), 인터 예측부(221) 및 인트라 예측부(222)에서 설명된 실시예들은 각각 디코딩 장치(300)의 필터링부(350), 인터 예측부(332) 및 인트라 예측부(331)에도 동일 또는 대응되도록 적용될 수 있다.

[80] 상술한 바와 같이 비디오 코딩을 수행함에 있어 압축 효율을 높이기 위하여 예측을 수행한다. 이를 통하여 코딩 대상 블록인 현재 블록에 대한 예측 샘플들을 포함하는 예측된 블록을 생성할 수 있다. 여기서 상기 예측된 블록은 공간 도메인(또는 픽셀 도메인)에서의 예측 샘플들을 포함한다. 상기 예측된 블록은 인코딩 장치 및 디코딩 장치에서 동일하게 도출되며, 상기 인코딩 장치는 원본 블록의 원본 샘플 값 자체가 아닌 상기 원본 블록과 상기 예측된 블록 간의 레지듀얼에 대한 정보(레지듀얼 정보)를 디코딩 장치로 시그널링함으로써 영상 코딩 효율을 높일 수 있다. 디코딩 장치는 상기 레지듀얼 정보를 기반으로 레지듀얼 샘플들을 포함하는 레지듀얼 블록을 도출하고, 상기 레지듀얼 블록과 상기 예측된 블록을 합하여 복원 샘플들을 포함하는 복원 블록을 생성할 수 있고, 복원 블록들을 포함하는 복원 픽처를 생성할 수 있다.

[81] 상기 레지듀얼 정보는 변환 및 양자화 절차를 통하여 생성될 수 있다. 예를 들어, 인코딩 장치는 상기 원본 블록과 상기 예측된 블록 간의 레지듀얼 블록을 도출하고, 상기 레지듀얼 블록에 포함된 레지듀얼 샘플들(레지듀얼 샘플 어레이)에 변환 절차를 수행하여 변환 계수들을 도출하고, 상기 변환 계수들에 양자화 절차를 수행하여 양자화된 변환 계수들을 도출하여 관련된 레지듀얼 정보를 (비트스트림을 통하여) 디코딩 장치로 시그널링할 수 있다. 여기서 상기

레지듀얼 정보는 상기 양자화된 변환 계수들의 값 정보, 위치 정보, 변환 기법, 변환 커널, 양자화 파라미터 등의 정보를 포함할 수 있다. 디코딩 장치는 상기 레지듀얼 정보를 기반으로 역양자화/역변환 절차를 수행하고 레지듀얼 샘플들(또는 레지듀얼 블록)을 도출할 수 있다. 디코딩 장치는 예측된 블록과 상기 레지듀얼 블록을 기반으로 복원 픽처를 생성할 수 있다. 인코딩 장치는 또한 이후 픽처의 인터 예측을 위한 참조를 위하여 양자화된 변환 계수들을 역양자화/역변환하여 레지듀얼 블록을 도출하고, 이를 기반으로 복원 픽처를 생성할 수 있다.

[82] 도 4는 신택스 엘리먼트(syntax element)를 인코딩하기 위한 CABAC(context-adaptive binary arithmetic coding)을 예시적으로 나타낸다.

[83] 예를 들어, CABAC의 부호화 과정은 인코딩 장치는 입력 신호가 이진값이 아닌 신택스 엘리먼트인 경우에는 상기 입력 신호의 값을 이진화(binarization)하여 입력 신호를 이진값로 변환할 수 있다. 또한, 상기 입력 신호가 이미 이진값인 경우(즉, 상기 입력 신호의 값이 이진값인 경우)에는 이진화가 수행되지 않고 바이패스(bypass)될 수 있다. 여기서, 이진값을 구성하는 각각의 이진수 0 또는 1을 빈(bin)이라고 할 수 있다. 예를 들어, 이진화된 후의 이진 스트링이 110인 경우, 1, 1, 0 각각을 하나의 빈이라고 한다. 하나의 신택스 엘리먼트에 대한 상기 빈(들)은 상기 신택스 엘리먼트의 값을 나타낼 수 있다.

[84] 이후, 상기 신택스 엘리먼트의 이진화된 빈들은 정규(regular) 부호화 엔진 또는 바이패스 부호화 엔진으로 입력될 수 있다.

[85] 인코딩 장치의 정규 부호화 엔진은 해당 빈에 대해 확률값을 반영하는 컨텍스트 모델(context model)을 할당할 수 있고, 할당된 컨텍스트 모델을 기반으로 해당 빈을 인코딩할 수 있다. 인코딩 장치의 상기 정규 부호화 엔진은 각 빈에 대한 인코딩을 수행한 뒤에 해당 빈에 대한 컨텍스트 모델을 갱신할 수 있다. 상술한 내용과 같이 인코딩되는 빈은 컨텍스트 코딩된 빈(context-coded bin)이라고 나타낼 수 있다.

[86] 컨텍스트 모델은 컨텍스트 코딩(정규 코딩)되는 빈 별로 할당 및 업데이트될 수 있으며, 컨텍스트 모델은 ctxIdx 또는 ctxInc를 기반으로 지시될 수 있다. ctxIdx는 ctxInc를 기반으로 도출될 수 있다. 예를 들어, 정규 코딩되는 빈들 각각에 대한 컨텍스트 모델을 가리키는 컨텍스트 인덱스(ctxIdx)는 context index increment(ctxInc) 및 context index offset(ctxIdxOffset)의 합으로 도출될 수 있다. 여기서, 상기 ctxInc는 각 빈 별로 다르게 도출될 수 있다. 상기 ctxIdxOffset는 상기 ctxIdx의 최소값(the lowest value)로 나타내어질 수 있다. 상기 ctxIdx의 최소값은 상기 ctxIdx의 초기값(initValue)으로 불릴 수도 있다. 상기 ctxIdxOffset은 일반적으로 다른 신택스 요소에 대한 컨텍스트 모델들과의 구분을 위하여 이용되는 값일 수 있으며, 하나의 신택스 요소에 대한 컨텍스트 모델은 ctxInc를 기반으로 구분 또는 도출될 수 있다.

[87] 한편, 상기 신택스 엘리먼트의 이진화된 빈들이 상기 바이패스 부호화 엔진에

입력되는 경우에는 다음과 같이 코딩될 수 있다. 예를 들어, 인코딩 장치의 바이패스 부호화 엔진은 입력된 빈에 대해 확률을 추정하는 절차와 부호화 후에 상기 빈에 적용한 확률 모델을 갱신하는 절차를 생략한다. 바이패스 인코딩이 적용되는 경우, 인코딩 장치는 콘텍스트 모델을 할당하는 대신 균일한 확률 분포를 적용해 입력되는 빈을 인코딩할 수 있고, 이를 통하여 인코딩 속도를 향상시킬 수 있다. 상술한 내용과 같이 인코딩되는 빈은 바이패스 빈(bypass bin)이라고 나타낼 수 있다.

[88] 엔트로피 인코딩 절차에서는 정규 코딩 엔진을 통해 인코딩을 수행할 것인지, 바이패스 코딩 엔진을 통해 인코딩을 수행할 것인지를 결정할 수 있고, 코딩 경로를 스위칭할 수 있다. 엔트로피 디코딩은 엔트로피 인코딩과 동일한 과정을 역순으로 수행할 수 있다.

[89] 예를 들어, 신텍스 엘리먼트가 컨텍스트 모델을 기반으로 디코딩되는 경우, 디코딩 장치는 비트스트림을 통하여 상기 신텍스 엘리먼트에 해당하는 빈을 수신할 수 있고, 상기 신텍스 엘리먼트와 디코딩 대상 블록 또는 주변 블록의 디코딩 정보 혹은 이전 단계에서 디코딩된 심볼/빈의 정보를 이용하여 컨텍스트 모델(context model)을 결정할 수 있고, 결정된 컨텍스트 모델에 따라 상기 수신된 빈(bin)의 발생 확률을 예측하여 빈의 산술 디코딩(arithmetic decoding)를 수행하여 상기 신텍스 엘리먼트의 값을 도출할 수 있다. 이후, 상기 결정된 컨텍스트 모델로 다음으로 디코딩되는 빈의 컨텍스트 모델이 업데이트될 수 있다.

[90] 또한, 예를 들어, 신텍스 엘리먼트가 바이패스 디코딩되는 경우, 디코딩 장치는 비트스트림을 통하여 상기 신텍스 엘리먼트에 해당하는 빈을 수신할 수 있고, 균일한 확률 분포를 적용해 입력되는 빈을 디코딩할 수 있다. 이 경우, 디코딩 장치는 신텍스 엘리먼트의 컨텍스트 모델을 도출하는 절차와 디코딩 이후에 상기 빈에 적용한 컨텍스트 모델을 갱신하는 절차는 생략될 수 있다.

[91] 상술한 바와 같이 레지듀얼 샘플들은 변환, 양자화 과정을 거쳐서 양자화된 변환 계수들로 도출될 수 있다. 양자화된 변환 계수들은 변환 계수들이라고도 불릴 수 있다. 이 경우 블록 내 변환 계수들은 레지듀얼 정보의 형태로 시그널링될 수 있다. 상기 레지듀얼 정보는 레지듀얼 코딩 신텍스를 포함할 수 있다. 즉, 인코딩 장치는 레지듀얼 정보로 레지듀얼 코딩 신텍스를 구성하고 이를 인코딩하여 비트스트림 형태로 출력할 수 있고, 디코딩 장치는 비트스트림으로부터 레지듀얼 코딩 신텍스를 디코딩하여 레지듀얼 (양자화된) 변환 계수들을 도출할 수 있다. 상기 레지듀얼 코딩 신텍스는 후술하는 바와 같이 해당 블록에 대하여 변환이 적용되었는지, 블록 내 마지막 유효 변환 계수의 위치가 어디인지, 서브블록 내 유효 변환 계수가 존재하는지, 유효 변환 계수의 크기/부호가 어떠한지 등을 나타내는 신텍스 엘리먼트들(syntax elements)을 포함할 수 있다.

[92] 예를 들어, (양자화된) 변환 계수(즉, 상기 레지듀얼 정보)는

last_sig_coeff_x_prefix, last_sig_coeff_y_prefix, last_sig_coeff_x_suffix, last_sig_coeff_y_suffix, coded_sub_block_flag, sig_coeff_flag, par_level_flag, abs_level_gt1_flag, abs_level_gt3_flag, abs_remainder, coeff_sign_flag, dec_abs_level, mts_idx 등의 신텍스 엘리먼트들(syntax elements)을 기반으로 인코딩 및/또는 디코딩될 수 있다. 레지듀얼 데이터 인코딩/디코딩과 관련된 신텍스 엘리먼트들은 다음의 표와 같이 나타낼 수 있다.

[93] [표1]

residual_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	**Descriptor**
if((mts_idx[x0][y0] > 0 || (cu_sbt_flag && log2TbWidth < 6 && log2TbHeight < 6)) && cIdx = = 0 && log2TbWidth > 4)	
log2TbWidth = 4	
else	
log2TbWidth = Min(log2TbWidth, 5)	
if(mts_idx[x0][y0] > 0 || (cu_sbt_flag && log2TbWidth < 6 && log2TbHeight < 6)) && cIdx = = 0 && log2TbHeight > 4)	
log2TbHeight = 4	
else	
log2TbHeight = Min(log2TbHeight, 5)	
last_sig_coeff_x_prefix	ae(v)
last_sig_coeff_y_prefix	ae(v)
if(last_sig_coeff_x_prefix > 3)	
last_sig_coeff_x_suffix	ae(v)
if(last_sig_coeff_y_prefix > 3)	
last_sig_coeff_y_suffix	ae(v)
log2SbSize = (Min(log2TbWidth, log2TbHeight) < 2 ? 1 : 2)	
numSbCoeff = 1 << (log2SbSize << 1)	
lastScanPos = numSbCoeff	
lastSubBlock = (1 << (log2TbWidth + log2TbHeight − 2 * log2SbSize)) − 1	
do {	
if(lastScanPos = = 0) {	
lastScanPos = numSbCoeff	
lastSubBlock− −	
}	
lastScanPos− −	
xS = DiagScanOrder[log2TbWidth − log2SbSize][log2TbHeight − log2SbSize][lastSubBlock][0]	
yS = DiagScanOrder[log2TbWidth − log2SbSize][log2TbHeight − log2SbSize][lastSubBlock][1]	

[94]

xC = (xS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][lastScanPos][0]	
yC = (yS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][lastScanPos][1]	
} while((xC != LastSignificantCoeffX) || (yC != LastSignificantCoeffY))	
numSigCoeff = 0	
QState = 0	
for(i = lastSubBlock; i >= 0; i− −) {	
startQStateSb = QState	
xS = DiagScanOrder[log2TbWidth − log2SbSize][log2TbHeight − log2SbSize][lastSubBlock][0]	
yS = DiagScanOrder[log2TbWidth − log2SbSize][log2TbHeight − log2SbSize][lastSubBlock][1]	
inferSbDcSigCoeffFlag = 0	
if((i < lastSubBlock) && (i > 0)) {	
coded_sub_block_flag[xS][yS]	ae(v)
inferSbDcSigCoeffFlag = 1	
}	
firstSigScanPosSb = numSbCoeff	
lastSigScanPosSb = −1	
remBinsPass1 = (log2SbSize < 2 ? 6 : 28)	
remBinsPass2 = (log2SbSize < 2 ? 2 : 4)	
firstPosMode0 = (i == lastSubBlock ? lastScanPos − 1 : numSbCoeff − 1)	
firstPosMode1 = −1	
firstPosMode2 = −1	
for(n = (i == firstPosMode0; n >= 0 && remBinsPass1 >= 3; 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 > 0 || !inferSbDcSigCoeffFlag)) {	

[95]

sig_coeff_flag[xC][yC]	ae(v)
remBinsPass1− −	
if(sig_coeff_flag[xC][yC])	
inferSbDcSigCoeffFlag = 0	
}	
if(sig_coeff_flag[xC][yC]) {	
numSigCoeff++	
abs_level_gt1_flag[n]	ae(v)
remBinsPass1− −	
if(abs_level_gt1_flag[n]) {	
par_level_flag[n]	ae(v)
remBinsPass1− −	
abs_level_gt3_flag[n]	ae(v)
remBinsPass1− −	
}	
if(lastSigScanPosSb == −1)	
lastSigScanPosSb = n	
firstSigScanPosSb = n	
}	

[96]

AbsLevelPass1[xC][yC] = sig_coeff_flag[xC][yC] + par_level_flag[n] + abs_level_gt1_flag[n] + 2 * abs_level_gt3_flag[n]	
if(dep_quant_enabled_flag)	
QState = QStateTransTable[QState][AbsLevelPass1[xC][yC] & 1]	
if(remBinsPass1 < 3)	
firstPosMode2 = n − 1	
}	
for(n = numSbCoeff − 1; n >= firstPosMode2; n− −) {	
xC = (xS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][0]	
yC = (yS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][1]	
if(abs_level_gt3_flag[n])	
abs_remainder[n]	ae(v)
AbsLevel[xC][yC] = AbsLevelPass1[xC][yC] + 2 * abs_remainder[n]	
}	
for(n = firstPosMode2; n >= 0; n− −) {	
xC = (xS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][0]	
yC = (yS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][1]	
dec_abs_level[n]	ae(v)
if(AbsLevel[xC][yC] > 0)	
firstSigScanPosSb = n	
if(dep_quant_enabled_flag)	
QState = QStateTransTable[QState][AbsLevel[xC][yC] & 1]	
}	

[97]

if(dep_quant_enabled_flag || !sign_data_hiding_enabled_flag)	
signHidden = 0	
else	
signHidden = (lastSigScanPosSb − firstSigScanPosSb > 3 ? 1 : 0)	
for(n = numSbCoeff − 1; n >= 0; n− −) {	
xC = (xS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][0]	
yC = (yS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][1]	
if(sig_coeff_flag[xC][yC] && (!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 << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][0]	
yC = (yS << log2SbSize) + DiagScanOrder[log2SbSize][log2SbSize][n][1]	
if(sig_coeff_flag[xC][yC])	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = (2 * AbsLevel[xC][yC] − (QState > 1 ? 1 : 0)) * (1 − 2 * coeff_sign_flag[n])	
QState = QStateTransTable[QState][par_level_flag[n]]	
} else {	

[98]

```
sumAbsLevel = 0
for( n = numSbCoeff − 1; n >= 0; n− − ) {
  xC = ( xS << log2SbSize ) +
       DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]
  yC = ( yS << log2SbSize ) +
       DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]
  if( sig_coeff_flag[ xC ][ yC ] ) {
    TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] =
      AbsLevel[ xC ][ yC ] * ( 1 − 2 * coeff_sign_flag[ n ] )
    if( signHidden ) {
      sumAbsLevel += AbsLevel[ xC ][ yC ]
      if( ( n == firstSigScanPosSb ) && ( sumAbsLevel % 2 ) == 1 ) )
        TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] =
          −TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ]
    }
  }
}
```

[99] 상술한 표 1을 참조하면 last_sig_coeff_x_prefix, last_sig_coeff_y_prefix, last_sig_coeff_x_suffix, last_sig_coeff_y_suffix, coded_sub_block_flag, sig_coeff_flag, abs_level_gt1_flag, par_level_flag, abs_level_gt3_flag, abs_remainder, dec_abs_level, 및/또는 coeff_sign_flag 가 인코딩/디코딩될 수 있다.

[100] 변환(및 양자화) 및 레지듀얼 코딩 절차에 관하여, CB(coding block)와 TB(Transform block)는 혼용될 수 있다. 예를 들어, CB에 대하여 레지듀얼 샘플들이 도출되고, 상기 레지듀얼 샘플들에 대한 변환 및 양자화를 통하여 (양자화된) 변환 계수들이 도출될 수 있음은 상술한 바와 같으며, 레지듀얼 코딩 절차를 통하여 상기 (양자화된) 변환 계수들의 위치, 크기, 부호 등을 효율적으로 나타내는 정보(예를 들어, 신텍스 엘리먼트들)이 생성되고 시그널링될 수 있다. 양자화된 변환 계수들은 간단히 변환 계수들이라고 불릴 수 있다. 일반적으로 CB가 최대 TB보다 크지 않은 경우, CB의 사이즈는 TB의 사이즈와 같을 수 있으며, 이 경우 변환(및 양자화) 및 레지듀얼 코딩되는 대상 블록은 CB 또는 TB라고 불릴 수 있다. 한편, CB가 최대 TB보다 큰 경우에는 변환(및 양자화) 및 레지듀얼 코딩되는 대상 블록은 TB라고 불릴 수 있다. 이하 레지듀얼 코딩에 관련된 신텍스 요소들이 변환 블록(TB) 단위로 시그널링되는 것으로 설명하나, 이는 예시로서 상기 TB는 코딩 블록(CB)과 혼용될 수 있음은 상술한 바와 같다.

[101] 일 실시예에서, 인코딩 장치는 신텍스 엘리먼트 last_sig_coeff_x_prefix, last_sig_coeff_y_prefix, last_sig_coeff_x_suffix 및 last_sig_coeff_y_suffix를 기반으로 변환 블록 내의 마지막 0이 아닌 변환 계수의 (x, y) 위치 정보를 인코딩할 수 있다. 보다 구체적으로, 상기 last_sig_coeff_x_prefix는 변환 블록 내 스캔 순서(scanning order)에서의 마지막(last) 유효 계수(significant coefficient)의 열 위치(column position)의 프리픽스(prefix)를 나타내고, 상기 last_sig_coeff_y_prefix는 상기 변환 블록 내 상기 스캔 순서(scanning order)에서의

마지막(last) 유효 계수(significant coefficient)의 행 위치(row position)의 프리픽스(prefix)를 나타내고, 상기 last_sig_coeff_x_suffix는 상기 변환 블록 내 상기 스캔 순서(scanning order)에서의 마지막(last) 유효 계수(significant coefficient)의 열 위치(column position)의 서픽스(suffix)를 나타내고, 상기 last_sig_coeff_y_suffix는 상기 변환 블록 내 상기 스캔 순서(scanning order)에서의 마지막(last) 유효 계수(significant coefficient)의 행 위치(row position)의 서픽스(suffix)를 나타낸다. 여기서, 유효 계수는 상기 0이 아닌 계수(non-zero coefficient)를 나타낼 수 있다. 또한, 상기 스캔 순서는 우상향 대각 스캔 순서일 수 있다. 또는 상기 스캔 순서는 수평 스캔 순서, 또는 수직 스캔 순서일 수 있다. 상기 스캔 순서는 대상 블록(CB, 또는 TB를 포함하는 CB)에 인트라/인터 예측이 적용되는지 여부 및/또는 구체적인 인트라/인터 예측 모드를 기반으로 결정될 수 있다.

[102] 그 다음, 인코딩 장치는 상기 변환 블록을 4x4 서브 블록(sub-block)들로 분할한 후, 각 4x4 서브 블록마다 1비트의 신택스 요소 coded_sub_block_flag를 사용해 현재 서브 블록 내에 0이 아닌 계수가 존재하는지 여부를 나타낼 수 있다.

[103] coded_sub_block_flag의 값이 0이면 더 이상 전송할 정보가 없으므로 인코딩 장치는 현재 서브 블록에 대한 부호화 과정을 종료할 수 있다. 반대로, coded_sub_block_flag의 값이 1이면 인코딩 장치는 sig_coeff_flag에 대한 부호화 과정을 계속해서 수행할 수 있다. 마지막 0이 아닌 계수를 포함하는 서브 블록은 coded_sub_block_flag에 대한 부호화가 불필요하고, 변환 블록의 DC 정보를 포함하고 있는 서브 블록은 0이 아닌 계수를 포함할 확률이 높으므로, coded_sub_block_flag는 부호화되지 않고 그 값이 1이라고 가정될 수 있다.

[104] 만약 coded_sub_block_flag의 값이 1이어서 현재 서브 블록 내에 0이 아닌 계수가 존재한다고 판단되는 경우, 인코딩 장치는 역으로 스캔된 순서에 따라 이진값을 갖는 sig_coeff_flag를 인코딩할 수 있다. 인코딩 장치는 스캔 순서에 따라 각각의 변환 계수에 대한 1비트 신택스 엘리먼트 sig_coeff_flag를 인코딩할 수 있다. 만약 현재 스캔 위치에서의 변환 계수의 값이 0이 아니면 sig_coeff_flag의 값은 1이 될 수 있다. 여기서, 마지막 0이 아닌 계수를 포함하고 있는 서브 블록의 경우, 마지막 0이 아닌 계수에 대해서는 sig_coeff_flag가 인코딩될 필요가 없으므로 상기 서브 블록에 대한 부호화 과정이 생략될 수 있다. sig_coeff_flag가 1인 경우에만 레벨 정보 부호화가 수행될 수 있으며, 레벨 정보 부호화 과정에는 네 개의 신택스 엘리먼트들이 사용될 수 있다. 보다 구체적으로, 각 sig_coeff_flag[xC][yC]는 현재 TB내 각 변환 계수 위치 (xC, yC)에서의 해당 변환 계수의 레벨(값)이 0이 아닌지(non-zero) 여부를 나타낼 수 있다. 일 실시예에서, 상기 sig_coeff_flag는 양자화된 변환 계수가 0이 아닌 유효 계수인지 여부를 나타내는 유효 계수 플래그의 신텍스 엘리먼트의 일 예시에 해당할 수 있다.

[105] sig_coeff_flag에 대한 부호화 이후의 남은 레벨 값은 아래의 수학식과 같이

도출될 수 있다. 즉, 부호화해야 할 레벨 값을 나타내는 신택스 요소 remAbsLevel은 아래의 수학식과 도출될 수 있다.

[106] [수식1]

$$remAbsLevel = |coeff| - 1$$

[107] 여기서, coeff는 실제 변환 계수값을 의미한다.

[108] 또한, abs_level_gt1_flag는 해당 스캐닝 위치(n)에서의 remAbsLevel'이 1보다 큰지 여부를 나타낼 수 있다. 예를 들어, abs_level_gt1_flag의 값이 0이면 해당 위치의 변환 계수의 절댓값(absolute value)은 1일 수 있다. 또한, 상기 abs_level_gt1_flag의 값이 1이면, 이후 부호화해야 할 레벨 값을 나타내는 상기 remAbsLevel은 아래의 수학식과 같이 도출될 수 있다.

[109] [수식2]

$$remAbsLevel = remAbsLevel - 1$$

[110] 또한, 상술한 수학식 2에 기재된 remAbsLevel의 least significant coefficient (LSB) 값은 par_level_flag을 통하여 아래의 수학식 3와 같이 인코딩될 수 있다.

[111] [수식3]

$$par\_level\_flag = remAbsLevel \,\&\, 1$$

[112] 여기서 par_level_flag[n]는 스캐닝 위치 n에서의 변환 계수 레벨(값)의 패리티(parity)를 나타낼 수 있다.

[113] par_level_flag 인코딩 후에 인코딩해야 할 변환 계수 레벨 값 remAbsLevel은 다음의 수학식과 같이 업데이트될 수 있다.

[114] [수식4]

$$remAbsLevel' = remAbsLevel >> 1$$

[115] abs_level_gt3_flag는 해당 스캐닝 위치(n)에서의 remAbsLevel'이 3보다 큰지 여부를 나타낼 수 있다. rem_abs_gt3_flag가 1인 경우에만 abs_remainder에 대한 인코딩이 수행될 수 있다. 실제 변환 계수값인 coeff와 각 신택스 요소들의 관계는 다음의 수학식과 같을 수 있다.

[116] [수식5]

$$|coeff| = sig\_coeff\_flag + abs\_level\_gt1\_flag + par\_level\_flag + 2 * (abs\_level\_gt3\_flag + abs\_remainder)$$

[117] 또한, 다음의 표는 상술한 수학식 5와 관련된 예시들을 나타낸다.

[118] [표2]

|coeff|	sig_coeff_flag	abs_level_gt1_flag	par_level_flag	abs_level_gt3_flag	abs_remainder / dec_abs_level
0	0				
1	1	0			
2	1	1	0		
3	1	1	1	0	
4	1	1	0	1	0
5	1	1	1	1	0
6	1	1	0	1	1
7	1	1	1	1	1
8	1	1	0	1	2
9	1	1	1	1	2
10	1	1	0	1	3
11	1	1	1	1	3
...	...	...	...		

[119] 여기서, |coeff|는 변환 계수 레벨(값)을 나타내며, 변환 계수에 대한 AbsLevel이라고 표시될 수도 있다. 또한, 각 계수의 부호는 1비트 심볼인 coeff_sign_flag를 이용하여 인코딩될 수 있다.

[120] 한편, 상술한 레지듀얼 정보는 transform_skip_flag를 더 포함할 수 있다. transform_skip_flag는 연관된 블록(associated block)에 변환이 생략되는지 여부를 나타낸다. 상기 transform_skip_flag는 변환 스킵 플래그의 신텍스 엘리먼트일 수 있다.

[121] 한편, 상술한 신텍스 엘리먼트들을 전송하는 실시예와 다른 예로, 레지듀얼 코딩을 위하여 변환 스킵이 적용되는지 여부에 따라 서로 상이한 레지듀얼 코딩 방식, 즉, 변환 스킵이 적용되는지 여부에 따라 상이한 레지듀얼 신텍스 엘리먼트들을 전송하는 실시예가 제안될 수 있다.

[122] 상술한 예에 따른 레지듀얼 코딩에 대한 신텍스 엘리먼트들은 다음의 표들과 같이 나타낼 수 있다.

[123] [표3]

transform_unit(x0, y0, tbWidth, tbHeight, treeType, subTuIndex, chType) {	Descriptor
transform_skip_flag[x0][y0][0]	ae(v)
if(!transform_skip_flag[x0][y0][0] || slice_ts_residual_coding_disabled_flag)	
residual_coding(x0, y0, Log2(tbWidth), Log2(tbHeight), 0)	
else	
residual_ts_coding(x0, y0, Log2(tbWidth), Log2(tbHeight), 0)	
}	
if(tu_cbf_cb[xC][yC] && treeType != DUAL_TREE_LUMA) {	
if(sps_transform_skip_enabled_flag && !BdpcmFlag[x0][y0][1] && wC <= MaxTsSize && hC <= MaxTsSize && !cu_sbt_flag)	
transform_skip_flag[xC][yC][1]	ae(v)
if(!transform_skip_flag[xC][yC][1] || slice_ts_residual_coding_disabled_flag)	
residual_coding(xC, yC, Log2(wC), Log2(hC), 1)	
else	
residual_ts_coding(xC, yC, Log2(wC), Log2(hC), 1)	
}	
if(tu_cbf_cr[xC][yC] && treeType != DUAL_TREE_LUMA && !(tu_cbf_cb[xC][yC] && tu_joint_cbcr_residual_flag[xC][yC])) {	
if(sps_transform_skip_enabled_flag && !BdpcmFlag[x0][y0][2] && wC <= MaxTsSize && hC <= MaxTsSize && !cu_sbt_flag)	
transform_skip_flag[xC][yC][2]	ae(v)
if(!transform_skip_flag[xC][yC][2] || slice_ts_residual_coding_disabled_flag)	
residual_coding(xC, yC, Log2(wC), Log2(hC), 2)	
else	
residual_ts_coding(xC, yC, Log2(wC), Log2(hC), 2)	
}	
}	

[124] [표4]

residual_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	Descriptor
if(sps_mts_enabled_flag && cu_sbt_flag && cIdx == 0 && log2TbWidth == 5 && log2TbHeight < 6)	
log2ZoTbWidth = 4	
else	
log2ZoTbWidth = Min(log2TbWidth, 5)	
if(sps_mts_enabled_flag && cu_sbt_flag && cIdx == 0 && log2TbWidth < 6 && log2TbHeight == 5)	
log2ZoTbHeight = 4	
else	
log2ZoTbHeight = Min(log2TbHeight, 5)	
if(log2TbWidth > 0)	
last_sig_coeff_x_prefix	ae(v)
if(log2TbHeight > 0)	
last_sig_coeff_y_prefix	ae(v)
if(last_sig_coeff_x_prefix > 3)	
last_sig_coeff_x_suffix	ae(v)
if(last_sig_coeff_y_prefix > 3)	
last_sig_coeff_y_suffix	ae(v)
log2TbWidth = log2ZoTbWidth	
log2TbHeight = log2ZoTbHeight	
remBinsPass1 = ((1 << (log2TbWidth + log2TbHeight)) * 7) >> 2	
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) {	

[125]

log2SbH = log2TbHeight	
log2SbW = 4 − log2SbH	
}	
numSbCoeff = 1 << (log2SbW + log2SbH)	
lastScanPos = numSbCoeff	
lastSubBlock = (1 << (log2TbWidth + log2TbHeight − (log2SbW + log2SbH))) − 1	
do {	
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]	

[126]

} while((xC != LastSignificantCoeffX) || (yC != LastSignificantCoeffY))	
if(lastSubBlock == 0 && log2TbWidth >= 2 && log2TbHeight >= 2 && !transform_skip_flag[x0][y0][cIdx] && lastScanPos > 0)	
LfnstDcOnly = 0	
if((lastSubBlock > 0 && log2TbWidth >= 2 && log2TbHeight >= 2) || (lastScanPos > 7 && (log2TbWidth == 2 || log2TbWidth == 3) && log2TbWidth == log2TbHeight))	
LfnstZeroOutSigCoeffFlag = 0	
if((lastSubBlock > 0 || lastScanPos > 0) && cIdx == 0)	
MtsDcOnly = 0	
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) {	
coded_sub_block_flag[xS][yS]	ae(v)
inferSbDcSigCoeffFlag = 1	
}	
if(coded_sub_block_flag[xS][yS] && (xS > 3 || yS > 3) && cIdx == 0)	
MtsZeroOutSigCoeffFlag = 0	
firstSigScanPosSb = numSbCoeff	
lastSigScanPosSb = −1	
firstPosMode0 = (i == lastSubBlock ? lastScanPos : numSbCoeff − 1)	
firstPosMode1 = firstPosMode0	

[127]

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)) {	
sig_coeff_flag[xC][yC]	ae(v)
remBinsPass1− −	
if(sig_coeff_flag[xC][yC])	
inferSbDcSigCoeffFlag = 0	
}	
if(sig_coeff_flag[xC][yC]) {	
abs_level_gtx_flag[n][0]	ae(v)
remBinsPass1− −	
if(abs_level_gtx_flag[n][0]) {	
par_level_flag[n]	ae(v)
remBinsPass1− −	
abs_level_gtx_flag[n][1]	ae(v)
remBinsPass1− −	
}	
if(lastSigScanPosSb == −1)	
lastSigScanPosSb = n	
firstSigScanPosSb = n	

[128]

}	
AbsLevelPass1[xC][yC] = sig_coeff_flag[xC][yC] + par_level_flag[n] + abs_level_gtx_flag[n][0] + 2 * abs_level_gtx_flag[n][1]	
if(ph_dep_quant_enabled_flag)	
QState = QStateTransTable[QState][AbsLevelPass1[xC][yC] & 1]	
firstPosMode1 = n − 1	
}	
for(n = firstPosMode0; n > firstPosMode1; n− −) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if(abs_level_gtx_flag[n][1])	
abs_remainder[n]	ae(v)
AbsLevel[xC][yC] = AbsLevelPass1[xC][yC] +2 * abs_remainder[n]	
}	
for(n = firstPosMode1; n >= 0; n− −) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if(coded_sub_block_flag[xS][yS])	
dec_abs_level[n]	ae(v)
if(AbsLevel[xC][yC] > 0) {	
if(lastSigScanPosSb == −1)	
lastSigScanPosSb = n	
firstSigScanPosSb = n	
}	
if(ph_dep_quant_enabled_flag)	
QState = QStateTransTable[QState][AbsLevel[xC][yC] & 1]	

[129]

}	
if(ph_dep_quant_enabled_flag || !pic_sign_data_hiding_enabled_flag)	
signHidden = 0	
else	
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) && (!signHidden || (n != firstSigScanPosSb)))	
coeff_sign_flag[n]	ae(v)
}	
if(ph_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][cIdx][xC][yC] = (2 * AbsLevel[xC][yC] − (QState > 1 ? 1 : 0)) * (1 − 2 * coeff_sign_flag[n])	
QState = QStateTransTable[QState][AbsLevel[xC][yC] & 1]	
} else {	

[130]

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][cIdx][xC][yC] = AbsLevel[xC][yC] * (1 − 2 * coeff_sign_flag[n])	
if(signHidden) {	
sumAbsLevel += AbsLevel[xC][yC]	
if((n == firstSigScanPosSb) && (sumAbsLevel % 2) == 1))	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = −TransCoeffLevel[x0][y0][cIdx][xC][yC]	
}	
}	
}	
}	
}	
}	

[131] [표5]

residual_ts_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	Descriptor
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)	
lastSubBlock = (1 << (log2TbWidth + log2TbHeight − (log2SbW + log2SbH))) − 1	
inferSbCbf = 1	
RemCcbs = ((1 << (log2TbWidth + log2TbHeight)) * 7) >> 2	
for(i =0; i <= lastSubBlock; i++) {	
xS = DiagScanOrder[log2TbWidth − log2SbW][log2TbHeight − log2SbH][i][0]	
yS = DiagScanOrder[log2TbWidth − log2SbW][log2TbHeight − log2SbH][i][1]	
if(i != lastSubBlock || !inferSbCbf)	
coded_sub_block_flag[xS][yS]	ae(v)
if(coded_sub_block_flag[xS][yS] && i < lastSubBlock)	
inferSbCbf = 0	
/* First scan pass */	
inferSbSigCoeffFlag = 1	
lastScanPosPass1 = −1	
for(n = 0; n <= numSbCoeff − 1 && RemCcbs >= 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 != numSbCoeff − 1 || !inferSbSigCoeffFlag)) {	

[132]

sig_coeff_flag[xC][yC]	ae(v)
RemCcbs− −	
if(sig_coeff_flag[xC][yC])	
inferSbSigCoeffFlag = 0	
}	
CoeffSignLevel[xC][yC] = 0	
if(sig_coeff_flag[xC][yC]) {	
coeff_sign_flag[n]	ae(v)
RemCcbs− −	
CoeffSignLevel[xC][yC] = (coeff_sign_flag[n] > 0 ? −1 : 1)	
abs_level_gtx_flag[n][0]	ae(v)
RemCcbs− −	
if(abs_level_gtx_flag[n][0]) {	
par_level_flag[n]	ae(v)
RemCcbs− −	
}	
}	
AbsLevelPass1[xC][yC] = sig_coeff_flag[xC][yC] + par_level_flag[n] + abs_level_gtx_flag[n][0]	
lastScanPosPass1 = n	
}	
/* Greater than X scan pass (numGtXFlags=5) */	
lastScanPosPass2 = −1	
for(n = 0; n <= numSbCoeff − 1 && RemCcbs >= 4; n++) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
AbsLevelPass2[xC][yC] = AbsLevelPass1[xC][yC]	
for(j = 1; j < 5; j++) {	

[133]

if(abs_level_gtx_flag[n][j − 1]) {	
abs_level_gtx_flag[n][j]	ae(v)
RemCcbs− −	
}	
AbsLevelPass2[xC][yC] += 2 * abs_level_gtx_flag[n][j]	
}	
lastScanPosPass2 = n	
}	
/* remainder scan pass */	
for(n = 0; n <= numSbCoeff − 1; n++) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if((n <= lastScanPosPass2 && AbsLevelPass2[xC][yC] >= 10) || (n > lastScanPosPass2 && n <= lastScanPosPass1 && AbsLevelPass1[xC][yC] >= 2) || (n > lastScanPosPass1 && coded_sub_block_flag[xS][yS]))	
abs_remainder[n]	ae(v)
if(n <= lastScanPosPass2)	
AbsLevel[xC][yC] = AbsLevelPass2[xC][yC] + 2 * abs_remainder[n]	
else if(n <= lastScanPosPass1)	
AbsLevel[xC][yC] = AbsLevelPass1[xC][yC] + 2 * abs_remainder[n]	
else { /* bypass */	

[134]

AbsLevel[xC][yC] = abs_remainder[n]	
if(abs_remainder[n])	
coeff_sign_flag[n]	ae(v)
}	
if(BdpcmFlag[x0][y0][cIdx] == 0 && n <= lastScanPosPass1) {	
absLeftCoeff = xC > 0 ? AbsLevel[xC − 1][yC] : 0	
absAboveCoeff = yC > 0 ? AbsLevel[xC][yC − 1] : 0	
predCoeff = Max(absLeftCoeff, absAboveCoeff)	
if(AbsLevel[xC][yC] == 1 && predCoeff > 0)	
AbsLevel[xC][yC] = predCoeff	
else if(AbsLevel[xC][yC] > 0 && AbsLevel[xC][yC] <= predCoeff)	
AbsLevel[xC][yC]− −	
}	
}	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = (1 − 2 * coeff_sign_flag[n]) * AbsLevel[xC][yC]	
}	
}	

[135] 본 실시예에 따르면, 표 3에 도시된 바와 같이 변환 스킵 플래그의 신텍스 엘리먼트 transform_skip_flag의 값에 따라 레지듀얼 코딩이 분기될 수 있다. 즉, 변환 스킵 플래그의 값을 기반으로(변환 스킵 여부를 기반으로) 레지듀얼 코딩을 위하여 상이한 신택스 엘리먼트가 사용될 수 있다. 변환 스킵이 적용되지 않은 경우(즉, 변환이 적용된 경우)에 사용되는 레지듀얼 코딩은 레귤러 레지듀얼 코딩(Regular Residual Coding, RRC)라고 불릴 수 있으며, 변환 스킵이 적용되지 않은 경우(즉, 변환이 적용되지 않은 경우)의 레지듀얼 코딩은 변환 스킵 레지듀얼 코딩(Transform Skip Residual Coding, TSRC)라고 불릴 수 있다. 상기 표 4는 transform_skip_flag의 값이 0 인 경우, 즉, 변환이 적용된 경우의 레지듀얼 코딩의 신텍스 엘리먼트를 나타낼 수 있고, 표 5는 transform_skip_flag의 값이 1 인 경우, 즉 변환이 적용되지 않은 경우의 레지듀얼 코딩의 신텍스 엘리먼트를 나타낼 수 있다.

[136] 예를 들어, 변환 블록의 변환 스킵 여부를 지시하는 변환 스킵 플래그가 파싱될 수 있고, 상기 변환 스킵 플래그가 1인지 여부가 판단될 수 있다. 상기 변환 스킵

플래그의 값이 1인 경우, 표 5에 도시된 바와 같이 변환 블록의 레지듀얼 계수에 대한 신텍스 엘리먼트들 sig_coeff_flag, coeff_sign_flag, abs_level_gtx_flag 및/또는 abs_remainder가 파싱 될 수 있고, 상기 신텍스 엘리먼트들을 기반으로 상기 레지듀얼 계수가 도출될 수 있다. 이 경우, 상기 신텍스 엘리먼트들은 순차적으로 파싱될 수도 있고, 파싱 순서가 변경될 수도 있다. 또한, 상기 abs_level_gtx_flag 는 abs_level_gt1_flag, abs_level_gt3_flag, abs_level_gt5_flag, abs_level_gt7_flag 및/또는 abs_level_gt9_flag을 나타낼 수 있다. 예를 들어, abs_level_gtx_flag[n][j]는 스캐닝 위치 n에서 변환 계수 레벨(또는 변환 계수 레벨을 우측으로 1만큼 쉬프팅한 값)의 절대값이 (j<<1)+1보다 큰지 여부를 나타내는 플래그일 수 있다. 상기 (j<<1)+1은, 경우에 따라서 제1 임계치, 제2 임계치 등 소정의 임계치로 대체될 수도 있다.

[137] 또한, 상기 변환 스킵 플래그의 값이 0인 경우, 표 4에 도시된 바와 같이 변환 블록의 레지듀얼 계수에 대한 신텍스 엘리먼트들 sig_coeff_flag, abs_level_gtx_flag, par_level_flag, abs_remainder, dec_abs_level, coeff_sign_flag 가 파싱될 수 있고, 상기 신텍스 엘리먼트들을 기반으로 상기 레지듀얼 계수가 도출될 수 있다. 이 경우, 상기 신텍스 엘리먼트들은 순차적으로 파싱될 수도 있고, 파싱 순서가 변경될 수도 있다. 또한, 상기 abs_level_gtx_flag 는 abs_level_gt1_flag 및/또는 abs_level_gt3_flag 을 나타낼 수 있다. 예를 들어, abs_level_gtx_flag[n][0]은 제1 변환 계수 레벨 플래그(abs_level_gt1_flag)의 일 예시일 수 있고, 상기 abs_level_gtx_flag[n][1]은 제2 변환 계수 레벨 플래그(abs_level_gt3_flag)의 일 예시일 수 있다.

[138] 한편, CABAC은 높은 성능을 제공하지만 처리량(throughput) 성능이 좋지 않다는 단점을 갖는다. 이는 CABAC의 정규 부호화 엔진으로 인한 것으로, 정규 부호화(즉, CABAC의 정규 부호화 엔진을 통한 인코딩)는 이전 빈(bin)의 부호화를 통해 업데이트된 확률 상태와 범위를 사용하기 때문에 높은 데이터 의존성을 보이며, 확률 구간을 읽고 현재 상태를 판단하는데 많은 시간이 소요될 수 있다. CABAC의 처리량 문제는 컨텍스트 코딩된 빈(context-coded bin)의 수를 제한함으로써 해결될 수 있다. 예를 들어, sig_coeff_flag, abs_level_gt1_flag, par_level_flag, abs_level_gt3_flag를 표현하기 위해 사용된 빈의 합이 해당 블록의 사이즈에 따른 개수로 제한될 수 있다. 일 예로, 해당 블록이 4x4 사이즈의 블록인 경우, 상기 sig_coeff_flag, abs_level_gt1_flag, par_level_flag, abs_level_gt3_flag 에 대한 빈들의 합은 32개로 제한될 수 있고, 해당 블록이 2x2 사이즈의 블록인 경우, 상기 sig_coeff_flag, abs_level_gt1_flag, par_level_flag, abs_level_gt3_flag에 대한 빈들의 합은 8개로 제한될 수 있다. 상기 빈들의 제한된 개수는 remBinsPass1으로 나타낼 수 있다. 또는, 일 예로, 보다 높은 CABAC 처리량을 위해, 컨텍스트 부호화 빈(context coded bin)의 개수가 코딩 대상 CG를 포함하는 블록(CB 또는 TB)에 대해 제한될 수 있다. 다시 말해, 컨텍스트 부호화 빈의 개수가 블록(CB 또는 TB) 단위로 제한될 수 있다. 예를 들어, 현재 블록의

사이즈가 16x16이면, 현재 CG와 상관 없이 현재 블록에 대한 컨텍스트 부호화 빈의 개수가 상기 현재 블록의 픽셀 개수의 1.75배, 즉, 448개로 제한될 수 있다.

[139] 이 경우, 인코딩 장치는 문맥 요소를 부호화하는데 제한된 개수의 문맥 부호화 빈을 모두 사용하면, 나머지 계수들을 CABAC을 사용하지 않고 후술하는 상기 계수들에 대한 이진화 방법을 통하여 이진화하고, 바이패스 인코딩을 수행할 수 있다. 다시 말해, 예를 들어, 4x4 CG 에 대하여 코딩된 컨텍스트 부호화 빈(context coded bin)의 수가 32, 또는 2x2 CG 에 대하여 코딩된 컨텍스트 부호화 빈의 수가 8이 되는 경우에는 더 이상 컨텍스트 부호화 빈으로 코딩되는 sig_coeff_flag, abs_level_gt1_flag, par_level_flag, abs_level_gt3_flag는 인코딩되지 않을 수 있고, 후술한 표 6과 같이 곧바로 dec_abs_level로 인코딩될 수 있다. 또는, 예를 들어, 4x4 블록에 대하여 코딩된 컨텍스트 부호화 빈(context coded bin)의 수가 전체 블록의 픽셀 개수의 1.75배, 즉, 28로 제한되는 경우, 더 이상 컨텍스트 부호화 빈으로 코딩되는 sig_coeff_flag, abs_level_gt1_flag, par_level_flag, abs_level_gt3_flag는 인코딩되지 않을 수 있고, 후술한 표 6과 같이 곧바로 dec_abs_level로 인코딩될 수 있다.

[140] [표6]

coeff	dec_abs_level
0	0
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
...	...

[141] dec_abs_level 를 기반으로 |coeff| 값이 도출될 수 있다. 이 경우, 변환 계수값인 |coeff| 는 다음의 수학식과 같이 도출될 수 있다.

[142] [수식6]

$$|coeff| = dec\_abs\_level$$

[143] 또한, 상기 coeff_sign_flag은 해당 스캐닝 위치(n)에서의 변환 계수 레벨의 부호(sign)을 나타낼 수 있다. 즉, 상기 coeff_sign_flag은 해당 스캐닝 위치(n)에서의 변환 계수의 부호(sign)을 나타낼 수 있다. 또한, 상기 mts_idx는 현재 변환 블록 내 레지듀얼 샘플들에 대하여 수평 방향 및 수직 방향으로 적용되는 변환 커널들을 나타낼 수 있다.

[144] 도 5는 4x4 블록 내 변환 계수들의 예시를 도시하는 도면이다.

[145] 도 5의 4x4 블록은 양자화된 계수들의 일 예를 나타낸다. 도 5에 도시된 블록은 4x4 변환 블록이거나, 또는 8x8, 16x16, 32x32, 64x64 변환 블록의 4x4 서브 블록일 수 있다. 도 5의 4x4 블록은 루마 블록 또는 크로마 블록을 나타낼 수 있다.

[146] 예를 들어, 도 5의 역 대각선 스캔되는 계수들에 대한 인코딩 결과는 다음의 표와 같을 수 있다.

[147] [표7]

scan_pos	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
coefficients	0	0	0	0	1	-1	0	2	0	3	-2	-3	4	6	-7	10
sig_coeff_flag	0	0	0	0	1	1	0	1	0	1	1	1	1	1		
abs_level_gt1_flag					0	0		1		1	1	1	1	1		
par_level_flag								0		1	0	1	0	0		
abs_level_gt3_flag													1	1		
abs_remainder													0	1		
dec_abs_level															7	10
coeff_sign_flag	0	0	0	0	0	1	0	0	0	0	1	1	0	0	1	0

[148] 상술한 표 7에서 scan_pos는 역 대각선 스캔에 따른 계수의 위치를 나타낸다. scan_pos 15는 4x4 블록에서 가장 먼저 스캔되는, 즉 우하단 코너의 변환 계수일 수 있고, scan_pos 0은 가장 나중에 스캔되는, 즉 좌상단 코너의 변환 계수일 수 있다. 한편 일 실시예에서, 상기 scan_pos는 스캔 위치라고 지칭될 수도 있다. 예를 들어, 상기 scan_pos 0은 스캔 위치 0이라고 지칭될 수 있다.

[149] 한편, 상술한 내용과 같이 인코딩 장치는 입력 신호가 이진값이 아닌 신텍스 엘리먼트인 경우에는 상기 입력 신호의 값을 이진화(binarization)하여 입력 신호를 이진값로 변환할 수 있다. 또한, 디코딩 장치는 상기 신텍스 엘리먼트를 디코딩하여 상기 신텍스 엘리먼트의 이진화된 값(즉, 이진화된 빈)을 도출할 수 있고, 상기 이진화된 값을 역 이진화하여 상기 신텍스 엘리먼트의 값을 도출할 수 있다. 상기 이진화 과정은 후술하는 트렁케이티드 라이스(Truncated Rice, TR) 이진화 프로세스(binarization process), k차 Exp-Golomb (k-th order Exp-Golomb, EGk) 이진화 프로세스(binarization process), k차 Limited Exp-Golomb (Limited k-th order Exp-Golomb, Limited EGk), 또는 고정 길이(Fixed-length, FL) 이진화

프로세스(binarization process) 등으로 수행될 수 있다. 또한, 역 이진화 과정은 상기 TR 이진화 프로세스, 상기 EGk 이진화 프로세스 또는 상기 FL 이진화 프로세스를 기반으로 수행되어 상기 신텍스 엘리먼트의 값을 도출하는 과정을 나타낼 수 있다.

[150] 예를 들어, 상기 TR 이진화 프로세스는 다음과 같이 수행될 수 있다.

[151] 상기 TR 이진화 프로세스의 입력(input)은 TR 이진화에 대한 요청과 신텍스 엘리먼트에 대한 cMax 및 cRiceParam 일 수 있다. 또한, 상기 TR 이진화 프로세스의 출력(output)은 빈 스트링에 대응하는 값 symbolVal 에 대한 TR 이진화일 수 있다.

[152] 구체적으로, 일 예로, 신텍스 엘리먼트에 대한 접미사(suffix) 빈 스트링이 존재하는 경우에는 상기 신텍스 엘리먼트에 대한 TR 빈 스트링은 접두사(prefix) 빈 스트링과 접미사 빈 스트링의 결합(concatenation)일 수 있고, 상기 접미사 빈 스트링이 존재하지 않는 경우에는 상기 신텍스 엘리먼트에 대한 상기 TR 빈 스트링은 상기 접두사 빈 스트링일 수 있다. 예를 들어, 상기 접두사 빈 스트링은 후술하는 바와 같이 도출될 수 있다.

[153] 상기 신텍스 엘리먼트에 대한 상기 symbolVal 의 접두사 값(prefix value)은 다음의 수학식과 같이 도출될 수 있다.

[154] [수식7]

$$prefixVal = symbolVal >> cRiceParam$$

[155] 여기서, prefixVal 은 상기 symbolVal 의 접두사 값을 나타낼 수 있다. 상기 신텍스 엘리먼트의 상기 TR 빈 스트링의 접두사(즉, 접두사 빈 스트링)는 후술하는 바와 같이 도출될 수 있다.

[156] 예를 들어, 상기 prefixVal이 cMax >> cRiceParam보다 작은 경우, 접두사 빈 스트링은 binIdx에 의해 인덱싱되는(indexed) 길이 prefixVal + 1의 비트 스트링(bit string)일 수 있다. 즉, 상기 prefixVal이 cMax >> cRiceParam보다 작은 경우, 상기 접두사 빈 스트링은 binIdx가 가리키는 prefixVal + 1 비트수의 비트스트링일 수 있다. prefixVal보다 작은 binIdx 에 대한 빈은 1과 동일할 수 있다. 또한, prefixVal와 동일한 binIdx 에 대한 빈은 0과 동일할 수 있다.

[157] 예를 들어, 상기 prefixVal에 대한 단항 이진화(unary binarization)로 도출되는 빈 스트링은 다음의 표와 같을 수 있다.

[158] [표8]

prefixVal	Bin string					
0	0					
1	1	0				
2	1	1	0			
3	1	1	1	0		
4	1	1	1	1	0	
5	1	1	1	1	1	0
...						
binIdx	0	1	2	3	4	5

[159] 한편, 상기 prefixVal이 cMax >> cRiceParam보다 작지 않은 경우, 상기 접두사 빈 스트링은 길이가 cMax >> cRiceParam 이고 모든 빈이 1인 비트 스트링일 수 있다.

[160] 또한, cMax 가 symbolVal 보다 크고, cRiceParam 이 0보다 큰 경우, TR 빈 스트링의 접미사 빈 스트링이 존재할 수 있다. 예를 들어, 상기 접미사 빈 스트링은 후술하는 바와 같이 도출될 수 있다.

[161] 상기 신텍스 엘리먼트에 대한 상기 symbolVal 의 접미사 값(suffix value)은 다음의 수학식과 같이 도출될 수 있다.

[162] [수식8]

$$suffixVal = symbolVal - ((prefixVal) << cRiceParam)$$

[163] 여기서, suffixVal 은 상기 symbolVal 의 접미사 값을 나타낼 수 있다.

[164] TR 빈 스트링의 접미사(즉, 접미사 빈 스트링)은 cMax 값이 (1 << cRiceParam)-1 인 suffixVal에 대한 FL 이진화 프로세스를 기반으로 도출될 수 있다.

[165] 한편, 입력 파라미터인 cRiceParam 의 값이 0이면, 상기 TR 이진화는 정확하게 트렁케이티드 단항 이진화(truncated unary binarization)일 수 있고, 항상 디코딩되는 신텍스 엘리멘트의 가능한 최대 값과 동일한 cMax 값이 사용될 수 있다.

[166] 또한, 예를 들어, 상기 EGk 이진화 프로세스는 다음과 같이 수행될 수 있다. ue(v) 로 코딩된 신텍스 엘리먼트는 Exp-Golomb 코딩된 신텍스 엘리먼트일 수 있다.

[167] 일 예로, 0차 Exp-Golomb (0-th order Exp-Golomb, EG0) 이진화 프로세스는 다음과 같이 수행될 수 있다.

[168] 상기 신텍스 엘리먼트에 대한 파싱 프로세스(parsing process)는 비트스트림의 현재 위치에서 시작하여 첫번째 논-제로(non-zero) 비트를 포함한 비트를 읽어

0과 같은 선행 비트 수를 세는 것(counting)으로 시작될 수 있다. 상기 과정은 다음의 표와 같이 나타낼 수 있다.

[169] [표9]

```
leadingZeroBits = −1
for( b = 0; !b; leadingZeroBits++ )
    b = read_bits( 1 )
```

[170] 또한, 변수 codeNum 은 다음의 수학식과 같이 도출될 수 있다.

[171] [수식9]

$$codeNum = 2^{leadingZeroBits} - 1 + read\_bits(leadingZeroBits)$$

[172] 여기서, read_bits(leadingZeroBits)에서 반환된 값, 즉, read_bits(leadingZeroBits)가 나타내는 값은 첫번째로 기록된 가장 중요한 비트(most significant bit)에 대한 언사인드 정수(unsigned integer)의 이진 표현(binary representation)으로 해석될 수 있다.

[173] 비트 스트링을 "접두사(prefix)" 비트와 "접미사(suffix)" 비트로 분리한 Exp-Golomb 코드의 구조는 다음의 표와 같이 나타낼 수 있다.

[174] [표10]

Bit string form	Range of codeNum
1	0
0 1 x_0	1..2
0 0 1 x_1 x_0	3..6
0 0 0 1 x_2 x_1 x_0	7..14
0 0 0 0 1 x_3 x_2 x_1 x_0	15..30
0 0 0 0 0 1 x_4 x_3 x_2 x_1 x_0	31..62
...	...

[175] "접두사" 비트는 leadingZeroBits 계산을 위하여 상술한 내용과 같이 파싱된 비트일 수 있고, 표 10에서 비트 스트링의 0 또는 1로 표시될 수 있다. 즉, 상술한 표 10의 0 또는 1로 개시된 비트 스트링은 접두사 비트 스트링을 나타낼 수 있다. "접미사" 비트는 codeNum의 계산에서 파싱되는 비트일 수 있고, 상술한 표 10에서 xi로 표시될 수 있다. 즉, 상술한 표 10의 xi로 개시된 비트 스트링은 접미사 비트 스트링을 나타낼 수 있다. 여기서, i는 0에서 LeadingZeroBits-1의 범위의 값일 수 있다. 또한, 각 xi는 0 또는 1과 동일할 수 있다.

[176] 상기 codeNum 에 할당되는 비트 스트링은 다음의 표와 같을 수 있다.

[177] [표11]

Bit string	codeNum
1	0
0 1 0	1
0 1 1	2
0 0 1 0 0	3
0 0 1 0 1	4
0 0 1 1 0	5
0 0 1 1 1	6
0 0 0 1 0 0 0	7
0 0 0 1 0 0 1	8
0 0 0 1 0 1 0	9
...	...

[178] 신텍스 엘리먼트의 디스크립터(descriptor)가 ue(v) 인 경우, 즉, 신텍스 엘리먼트가 ue(v)로 코딩된 경우, 상기 신텍스 엘리먼트의 값은 codeNum과 동일할 수 있다.

[179] 또한, 예를 들어, 상기 EGk 이진화 프로세스는 다음과 같이 수행될 수 있다.

[180] 상기 EGk 이진화 프로세스의 입력(input)은 EGk 이진화에 대한 요청일 수 있다. 또한, 상기 EGk 이진화 프로세스의 출력(output)은 빈 스트링에 대응하는 값 symbolVal 에 대한 EGk 이진화일 수 있다.

[181] symbolVal에 대한 EGk 이진화 프로세스의 비트 스트링은 다음과 같이 도출될 수 있다.

[182] [표12]

```
absV = Abs( symbolVal )
stopLoop = 0
do
    if( absV >= ( 1 << k ) ) {
        put( 1 )
        absV = absV - ( 1 << k )
        k++
    } else {
        put( 0 )
        while( k-- )
            put( ( absV >> k ) & 1 )
        stopLoop = 1
    }
while( !stopLoop )
```

[183] 상술한 표 12를 참조하면, put(X)의 각 콜(each call)을 통하여 이진값 X를 빈 스트링의 끝에 추가될 수 있다. 여기서, X는 0 또는 1 일 수 있다.

[184] 또한, 예를 들어, 상기 Limited EGk 이진화 프로세스는 다음과 같이 수행될 수 있다.

[185] 상기 Limited EGk 이진화 프로세스의 입력(input)은 Limited EGk 이진화에 대한 요청 및 라이스 파라미터 riceParam, 최댓값의 이진대수를 나타내는 변수인 log2TransformRange 및 최대 접두사 확장 길이를 나타내는 변수인 maxPreExtLen일 수 있다. 또한, 상기 Limited EGk 이진화 프로세스의 출력(output)은 빈 스트링에 대응하는 값 symbolVal에 대한 Limited EGk 이진화일 수 있다.

[186] symbolVal에 대한 Limited EGk 이진화 프로세스의 비트 스트링은 다음과 같이 도출될 수 있다.

[187] [표13]

```
codeValue = symbolVal >> riceParam
PrefixExtensionLength = 0
while( ( PrefixExtensionLength < maxPrefixExtensionLength ) &&
        ( codeValue > ( ( 2 << PrefixExtensionLength ) - 2 ) ) ) {
    PrefixExtensionLength++
    put( 1 )
}
if( PrefixExtensionLength == maxPrefixExtensionLength )
    escapeLength = log2TransformRange
else {
    escapeLength = PrefixExtensionLength + riceParam
    put( 0 )
}
symbolVal = symbolVal - ( ( ( 1 << PrefixExtensionLength ) - 1 ) << riceParam )
while( ( escapeLength-- ) > 0 )
    put( ( symbolVal >> escapeLength ) & 1 )
```

[188] 또한, 예를 들어, 상기 FL 이진화 프로세스는 다음과 같이 수행될 수 있다.

[189] 상기 FL 이진화 프로세스의 입력(input)은 FL 이진화에 대한 요청 및 상기 신텍스 엘리먼트에 대한 cMax일 수 있다. 또한, 상기 FL 이진화 프로세스의 출력(output)은 빈 스트링에 대응하는 값 symbolVal 에 대한 FL 이진화일 수 있다.

[190] FL 이진화는 심볼값 symbolVal의 고정길이인 비트수를 갖는 비트 스트링을 사용하여 구성될 수 있다. 여기서, 상기 고정 길이 비트는 부호없는 정수 비트 스트링(unsigned integer bit string)일 수 있다. 즉, FL 이진화를 통하여 심볼값 symbolVal 에 대한 비트 스트링이 도출될 수 있고, 상기 비트스트링의 비트길이(즉, 비트수)는 고정 길이일 수 있다.

[191] 예를 들어, 상기 고정 길이는 다음의 수학식과 같이 도출될 수 있다.

[192] [수식10]

$$fixedLength = Ceil(Log2(cMax + 1))$$

[193] FL 이진화에 대한 빈들의 인덱싱은 최상위 비트에서 최하위 비트 순서로 증가하는 값을 사용하는 방식일 수 있다. 예를 들어, 상기 최상위 비트와 관련된 빈 인덱스는 binIdx = 0 일 수 있다.

[194] 한편, 예를 들어, 상기 레지듀얼 정보 중 신텍스 엘리먼트 abs_remainder 에 대한 이진화 프로세스는 다음과 같이 수행될 수 있다.

[195] 상기 abs_remainder 에 대한 이진화 프로세스의 입력은 신텍스 엘리먼트 abs_remainder[n] 의 이진화에 대한 요청, 색상 성분(colour component) cIdx, 루마 위치 (x0, y0) 일 수 있다. 상기 루마 위치 (x0, y0)는 픽처의 좌상단 루마 샘플을 기준으로 하는 현재 루마 변환 블록의 좌상단 샘플을 가리킬 수 있다.

[196] 상기 abs_remainder 에 대한 이진화 프로세스의 출력(output)은 상기 abs_remainder 의 이진화(즉, 상기 abs_remainder 의 이진화된 빈 스트링)일 수

있다. 상기 이진화 프로세스를 통하여 상기 abs_remainder 에 대한 가용 빈 스트링들이 도출될 수 있다.

[197] 먼저, abs_remainder[n] 에 대한 lastAbsRemainder 및 lastRiceParam 이 다음과 같이 도출될 수 있다. 여기서, 상기 lastAbsRemainder 는 상기 abs_remainder[n] 이전에 도출된 abs_remainder 의 값을 나타낼 수 있고, 상기 lastRiceParam 는 상기 abs_remainder[n] 이전에 도출된 abs_remainder 에 대한 라이스 파라미터 cRiceParam 를 나타낼 수 있다.

[198] 예를 들어, 상기 abs_remainder[n] 에 대한 lastAbsRemainder 및 lastRiceParam 를 도출하는 프로세스가 현재 서브 블록에 대해 처음 호출된 경우, 즉, 상기 현재 서브 블록의 변환 계수들 중 스캐닝 순서 상 첫번째 순서의 변환 계수에 대한 abs_remainder[n]의 프로세스가 수행되는 경우, 상기 lastAbsRemainder 및 상기 lastRiceParam은 모두 0으로 설정될 수 있다.

[199] 또한, 상기 경우가 아닌 경우, 즉, 상기 프로세스가 현재 서브 블록 대해 처음으로 호출된 경우가 아니면, 상기 lastAbsRemainder 및 상기 lastRiceParam은 각각의 마지막 호출에서 도출된 abs_remainder[n] 및 cRiceParam의 값과 동일하게 설정될 수 있다. 즉, 상기 lastAbsRemainder 는 현재 코딩되는 abs_remainder[n] 이전에 코딩된 abs_remainder[n] 와 동일한 값으로 도출될 수 있고, 상기 lastRiceParam 는 현재 코딩되는 abs_remainder[n] 이전에 코딩된 abs_remainder[n] 에 대한 cRiceParam 와 동일한 값으로 도출될 수 있다.

[200] 이후, 현재 코딩되는 abs_remainder[n] 에 대한 라이스 파라미터 cRiceParam 은 상기 lastAbsRemainder 및 상기 lastRiceParam 을 기반으로 도출될 수 있다. 예를 들어, 현재 코딩되는 abs_remainder[n] 에 대한 라이스 파라미터 cRiceParam 은 다음의 수학식과 같이 도출될 수 있다.

[201] [수식11]

$$cRiceParam = Min(lastRiceParam + ((lastAbsRemainder > (3 * (1 << lastRiceParam))) ? 1 : 0), 3)$$

[202] 또한, 예를 들어, 현재 코딩되는 abs_remainder[n] 에 대한 cMax는 상기 라이스 파라미터 cRiceParam 를 기반으로 도출될 수 있다. 상기 cMax는 다음의 수학식과 같이 도출될 수 있다.

[203] [수식12]

$$cMax = 6 << cRiceParam$$

[204] 또는, 예를 들어, 현재 블록의 변환 스킵 여부를 기반으로 상기 라이스 파라미터 cRiceParam이 결정될 수 있다. 즉, 현재 CG를 포함하는 현재 TB에 대해 변환이 적용되지 않는 경우, 다시 말해, 상기 현재 CG를 포함하는 상기 현재 TB에 대하여 변환 스킵(transform skip)이 적용되는 경우, 상기 라이스 파라미터 cRiceParam은 1로 도출될 수 있다. 또는, 상기 현재 CG를 포함하는 상기 현재

TB에 대해 변환이 적용되는 경우, 다시 말해, 상기 현재 CG를 포함하는 상기 현재 TB에 대하여 변환 스킵이 적용되지 않는 경우, 상기와 같이 현재 코딩되는 abs_remainder[n] 에 대한 라이스 파라미터 cRiceParam 은 이전에 코딩된 abs_remainder[n] 에 대한 cRiceParam과 동일한 값으로 도출될 수 있다.

[205] 한편, 상기 abs_remainder 에 대한 이진화, 즉, 상기 abs_remainder 에 대한 빈 스트링은 접미사(suffix) 빈 스트링이 존재하는 경우에는 접두사(prefix) 빈 스트링과 접미사 빈 스트링의 결합(concatenation)일 수 있다. 또한, 상기 접미사 빈 스트링이 존재하지 않는 경우에는 상기 abs_remainder 에 대한 상기 빈 스트링은 상기 접두사 빈 스트링일 수 있다.

[206] 예를 들어, 상기 접두사 빈 스트링은 후술하는 바와 같이 도출될 수 있다.

[207] 상기 abs_remainder[n] 의 접두사 값(prefix value) prefixVal은 다음의 수학식과 같이 도출될 수 있다.

[208] [수식13]

$$prefixVal = Min(cMax, abs\_remainder[n])$$

[209] 상기 abs_remainder[n] 의 상기 빈 스트링의 접두사(즉, 접두사 빈 스트링)는 상기 cMax 및 상기 cRiceParam 을 입력으로 사용하는 상기 prefixVal 에 대한 TR 이진화 프로세스를 통하여 도출될 수 있다.

[210] 상기 접두사 빈 스트링이 모든 비트가 1이고 비트 길이가 6인 비트 스트링과 동일하면 상기 abs_remainder[n] 의 상기 빈 스트링의 접미사 빈 스트링이 존재할 수 있고, 후술하는 바와 같이 도출될 수 있다.

[211] 상기 abs_remainder 의 접미사 값(suffix value) suffixVal은 다음의 수학식과 같이 도출될 수 있다.

[212] [수식14]

$$suffixVal = abs\_remainder[n] - cMax$$

[213] 상기 abs_remainder 의 상기 빈 스트링의 접미사 빈 스트링은 k 가 cRiceParam+1 로 설정되고, riceParam은 cRiceParam으로 설정되고, log2TransformRange는 15로 설정되고, maxPreExtLen은 11로 설정되는 상기 suffixVal 에 대한 Limited EGk 이진화 프로세스를 통하여 도출될 수 있다.

[214] 한편, 예를 들어, 상기 레지듀얼 정보 중 신텍스 엘리먼트 dec_abs_level 에 대한 이진화 프로세스는 다음과 같이 수행될 수 있다.

[215] 상기 dec_abs_level 에 대한 이진화 프로세스의 입력은 신텍스 엘리먼트 dec_abs_level[n] 의 이진화에 대한 요청, 색상 성분(colour component) cIdx, 루마 위치 (x0, y0), 현재 계수 스캔 위치 (xC, yC), 변환 블록의 폭의 이진대수(binary logarithm)인 log2TbWidth 및 변환 블록의 높이의 이진대수인 log2TbHeight 일 수 있다. 상기 루마 위치 (x0, y0)는 픽처의 좌상단 루마 샘플을 기준으로 하는 현재

루마 변환 블록의 좌상단 샘플을 가리킬 수 있다.

[216] 상기 dec_abs_level 에 대한 이진화 프로세스의 출력(output)은 상기 dec_abs_level 의 이진화(즉, 상기 dec_abs_level 의 이진화된 빈 스트링)일 수 있다. 상기 이진화 프로세스를 통하여 상기 dec_abs_level 에 대한 가용 빈 스트링들이 도출될 수 있다.

[217] 상기 dec_abs_level[n] 에 대한 라이스 파라미터 cRiceParam 은 상기 색상 성분 cIdx 및 루마 위치 (x0, y0), 현재 계수 스캔 위치 (xC, yC), 변환 블록의 폭의 이진대수인 log2TbWidth 및 변환 블록의 높이의 이진대수인 log2TbHeight 을 입력으로 수행되는 라이스 파라미터 도출 과정을 통하여 도출될 수 있다. 상기 라이스 파라미터 도출 과정에 대한 구체적인 설명은 후술한다.

[218] 또한, 예를 들어, 상기 dec_abs_level[n] 에 대한 cMax는 상기 라이스 파라미터 cRiceParam 를 기반으로 도출될 수 있다. 상기 cMax는 다음의 수학식과 같이 도출될 수 있다.

[219] [수식15]

$$cMax = 6 << cRiceParam$$

[220] 한편, 상기 dec_abs_level[n] 에 대한 이진화, 즉, 상기 dec_abs_level[n] 에 대한 빈 스트링은 접미사(suffix) 빈 스트링이 존재하는 경우에는 접두사(prefix) 빈 스트링과 접미사 빈 스트링의 결합(concatenation)일 수 있다. 또한, 상기 접미사 빈 스트링이 존재하지 않는 경우에는 상기 dec_abs_level[n] 에 대한 상기 빈 스트링은 상기 접두사 빈 스트링일 수 있다.

[221] 예를 들어, 상기 접두사 빈 스트링은 후술하는 바와 같이 도출될 수 있다.

[222] 상기 dec_abs_level[n] 의 접두사 값(prefix value) prefixVal은 다음의 수학식과 같이 도출될 수 있다.

[223] [수식16]

$$prefixVal = Min(cMax, dec\_abs\_level[n])$$

[224] 상기 dec_abs_level[n] 의 상기 빈 스트링의 접두사(즉, 접두사 빈 스트링)는 상기 cMax 및 상기 cRiceParam 을 입력으로 사용하는 상기 prefixVal 에 대한 TR 이진화 프로세스를 통하여 도출될 수 있다.

[225] 상기 접두사 빈 스트링이 모든 비트가 1이고 비트 길이가 6인 비트 스트링과 동일하면 상기 dec_abs_level[n] 의 상기 빈 스트링의 접미사 빈 스트링이 존재할 수 있고, 후술하는 바와 같이 도출될 수 있다.

[226] 상기 dec_abs_level[n] 에 대한 라이스 파라미터 도출 과정은 다음과 같을 수 있다.

[227] 상기 라이스 파라미터 도출 과정의 입력은 색상 성분 인덱스(colour component

index) cIdx, 루마 위치 (x0, y0), 현재 계수 스캔 위치 (xC, yC), 변환 블록의 폭의 이진대수(binary logarithm)인 log2TbWidth 및 변환 블록의 높이의 이진대수 인 log2TbHeight 일 수 있다. 상기 루마 위치 (x0, y0)는 픽처의 좌상단 루마 샘플을 기준으로 하는 현재 루마 변환 블록의 좌상단 샘플을 가리킬 수 있다. 또한, 상기 라이스 파라미터 도출 과정의 출력은 상기 라이스 파라미터 cRiceParam 일 수 있다.

[228] 예를 들어, 주어진 신텍스 엘리먼트들 sig_coeff_flag[x][y] 과 상기 컴포넌트 인덱스 cIdx, 상기 좌상단 루마 위치 (x0, y0)을 갖는 변환 블록에 대한 배열 AbsLevel[x][y] 을 기반으로 변수 locSumAbs 는 다음의 표에 개시된 슈도 코드(pseudo code)와 같이 도출될 수 있다.

[229] [표14]

```
locSumAbs = 0
if( xC < (1 << log2TbWidth) − 1 ) {
  locSumAbs += AbsLevel[ xC + 1 ][ yC ] − sig_coeff_flag[ xC + 1 ][ yC ]
  if( xC < (1 << log2TbWidth) − 2 )
    locSumAbs += AbsLevel[ xC + 2 ][ yC ] − sig_coeff_flag[ xC + 2 ][ yC ]
  if( yC < (1 << log2TbHeight) − 1 )
    locSumAbs += AbsLevel[ xC + 1 ][ yC + 1 ] − sig_coeff_flag[ xC + 1 ][ yC + 1 ]
}
if( yC < (1 << log2TbHeight) − 1 ) {
  locSumAbs += AbsLevel[ xC ][ yC + 1 ] − sig_coeff_flag[ xC ][ yC + 1 ]
  if( yC < (1 << log2TbHeight) − 2 )
    locSumAbs += AbsLevelPass1 [ xC ][ yC + 2 ] − sig_coeff_flag[ xC ][ yC + 2 ]
}
if( locSumAbs > 31 )
  locSumAbs = 31
```

[230] 상기 라이스 파라미터 cRiceParam 는 다음과 같이 도출될 수 있다.

[231] 예를 들어, 상기 라이스 파라미터 cRiceParam 는 상기 도출된 변수 locSumAbs 및 변수 s 를 기반으로 도출될 수 있다. 상기 변수 s 는 Max(0, QState - 1) 로 설정될 수 있다. 즉, 상기 s 는 0 과 QState - 1 중 최대값으로 설정될 수 있다.

[232] 상기 변수 locSumAbs 및 상기 변수 s 를 기반으로 도출되는 상기 라이스 파라미터 cRiceParam 및 ZeroPos[n] 는 다음의 표와 같을 수 있다.

[233] [표15]

s	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	1	1	1	1	1	1	1	2	2
0	ZeroPos[n]	0	0	0	0	0	1	2	2	2	2	2	2	4	4	4	4
1	ZeroPos[n]	1	1	1	1	2	3	4	4	4	6	6	6	8	8	8	8
2	ZeroPos[n]	1	1	2	2	2	3	4	4	4	6	6	6	8	8	8	8
	locSumAbs	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	cRiceParam	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3
0	ZeroPos[n]	4	4	4	4	4	4	4	8	8	8	8	8	16	16	16	16
1	ZeroPos[n]	4	4	12	12	12	12	12	12	12	12	16	16	16	16	16	16
2	ZeroPos[n]	8	8	12	12	12	12	12	12	12	16	16	16	16	16	16	16

[234] 예를 들어, 상술한 표 15를 참조하면 상기 locSumAbs 가 6 이하인 경우, 상기 cRiceParam 은 0으로 설정될 수 있고, 상기 locSumAbs 가 7 이상이고 13 이하인

경우, 상기 cRiceParam 은 1로 설정될 수 있고, 상기 locSumAbs 가 14 이상이고 27 이하인 경우, 상기 cRiceParam 은 2로 설정될 수 있고, 상기 locSumAbs 28이상인 경우, 상기 cRiceParam 은 3으로 설정될 수 있다.

[235] 또한, 상술한 표 15를 참조하면 상기 s 가 0 이고, 상기 locSumAbs 가 4 이하인 경우, 상기 ZeroPos[n] 은 0으로 설정될 수 있고, 상기 s 가 0 이고, 상기 locSumAbs 가 5인 경우, 상기 ZeroPos[n] 은 1로 설정될 수 있고, 상기 s 가 0 이고, 상기 locSumAbs 가 6 이상이고 11 이하인 경우, 상기 ZeroPos[n] 은 2로 설정될 수 있고, 상기 s 가 0 이고, 상기 locSumAbs 가 12 이상이고 22 이하인 경우, 상기 ZeroPos[n] 은 4로 설정될 수 있고, 상기 s 가 0 이고, 상기 locSumAbs 가 23 이상이고 27 이하인 경우, 상기 ZeroPos[n] 은 8로 설정될 수 있고, 상기 s 가 0 이고, 상기 locSumAbs 가 28 이상인 경우, 상기 ZeroPos[n] 은 16으로 설정될 수 있다. 또한, 상술한 표 15를 참조하면 상기 s 가 1 이고, 상기 locSumAbs 가 3 이하인 경우, 상기 ZeroPos[n] 은 1로 설정될 수 있고, 상기 s 가 1 이고, 상기 locSumAbs 가 4인 경우, 상기 ZeroPos[n] 은 2로 설정될 수 있고, 상기 s 가 1 이고, 상기 locSumAbs 가 5인 경우, 상기 ZeroPos[n] 은 3으로 설정될 수 있고, 상기 s 가 1 이고, 상기 locSumAbs 가 6 이상이고 8 이하인 경우, 상기 ZeroPos[n] 은 4로 설정될 수 있고, 상기 s 가 1 이고, 상기 locSumAbs 가 9 이상이고 11 이하인 경우, 상기 ZeroPos[n] 은 6으로 설정될 수 있고, 상기 s 가 1 이고, 상기 locSumAbs 가 12이상이고 15 이하인 경우, 상기 ZeroPos[n] 은 8로 설정될 수 있고, 상기 s 가 1 이고, 상기 locSumAbs 가 16이상이고 17 이하인 경우, 상기 ZeroPos[n] 은 4로 설정될 수 있고, 상기 s 가 1 이고, 상기 locSumAbs 가 18이상이고 25 이하인 경우, 상기 ZeroPos[n] 은 12로 설정될 수 있고, 상기 locSumAbs 가 26이상이고 31 이하인 경우, 상기 ZeroPos[n] 은 16으로 설정될 수 있다. 또한, 상술한 표 15를 참조하면 상기 s 가 2 이고, 상기 locSumAbs 가 1 이하인 경우, 상기 ZeroPos[n] 은 1로 설정될 수 있고, 상기 s 가 2 이고, 상기 locSumAbs 가 2 이상이고 4이하인 경우, 상기 ZeroPos[n] 은 2로 설정될 수 있고, 상기 s 가 2 이고, 상기 locSumAbs 가 5인 경우, 상기 ZeroPos[n] 은 3으로 설정될 수 있고, 상기 s 가 2 이고, 상기 locSumAbs 가 6 이상이고 8이하인 경우, 상기 ZeroPos[n] 은 4로 설정될 수 있고, 상기 s 가 2 이고, 상기 locSumAbs 가 9 이상이고 11이하인 경우, 상기 ZeroPos[n] 은 6으로 설정될 수 있고, 상기 s 가 2 이고, 상기 locSumAbs 가 12 이상이고 17이하인 경우, 상기 ZeroPos[n] 은 8로 설정될 수 있고, 상기 s 가 2 이고, 상기 locSumAbs 가 18 이상이고 24이하인 경우, 상기 ZeroPos[n] 은 12로 설정될 수 있고, 상기 s 가 2 이고, 상기 locSumAbs 가 25이상인 경우, 상기 ZeroPos[n] 은 16으로 설정될 수 있다.

[236] 또는, 예를 들어, ZeroPos[n]은 다음의 수식과 같이 도출될 수 있으며, 여기서, 후술하는 수식에 포함되는 cRiceParam은 표 15를 참조하여 도출될 수 있다.

[237] [수식17]

$$ZeroPos[n]= (QState < 2? 1 : 2) << cRiceParam$$

[238] 또한, 상기 dec_abs_level[n] 의 접미사 값(suffix value) suffixVal은 다음의 수학식과 같이 도출될 수 있다.

[239] [수식18]

$$suffixVal = dec\_abs\_level[\ n\ ] - cMax$$

[240] 상기 dec_abs_level[n] 의 상기 빈 스트링의 접미사 빈 스트링은 k 가 cRiceParam+1 로 설정되고, riceParam은 cRiceParam으로 설정되고, log2TransformRange는 15로 설정되고, maxPreExtLen은 11로 설정되는 상기 suffixVal 에 대한 Limited EGk 이진화 프로세스를 통하여 도출될 수 있다.

[241] 또한, 본 문서는 양자화된 예측 레지듀얼을 나타내는 변환 스킵 레벨(즉, 공간 도메인에서의 레지듀얼)의 통계 및 신호 특성을 레지듀얼 코딩에 적응시키기 위하여 기존 레지듀얼 코딩 방안에서 후술하는 내용을 수정하는 방안을 제안한다.

[242] 스캐닝 순서 : 예를 들어, TB 블록 내의 서브 블록 및 서브 블록 내의 레지듀얼 계수에 대한 스캐닝 순서는 우하단에서 좌상단으로 이동하는 대각 스캔 순서일 수 있다. 즉, TB 블록 내의 서브 블록 및 서브 블록 내의 레지듀얼 계수에 대한 스캐닝 순서는 우하단에서 좌상단 방향으로 스캔하는 대각 스캔 순서일 수 있다. 또는, 예를 들어, TB 블록 내의 서브 블록 및 서브 블록 내의 레지듀얼 계수에 대한 스캐닝 순서는 좌상단에서 우하단으로 이동하는 대각 스캔 순서일 수 있다. 즉, TB 블록 내의 서브 블록 및 서브 블록 내의 레지듀얼 계수에 대한 스캐닝 순서는 좌상단에서 우하단 방향으로 스캔하는 대각 스캔 순서일 수 있다.

[243] 마지막 논 제로(non-zero) 변환 계수 위치 없음: 레지듀얼 신호(즉, 레지듀얼 샘플)는 예측 후, 공간적 레지듀얼을 반영하고 변환 스킵(transform skip)으로 변환에 의한 에너지 압축이 수행되지 않기 때문에, 후행하는 0에 대한 높은 확률 또는 변환 블록의 우하단에 있는 중요하지 않은 레벨은 더 이상 발생하지 않을 수 있다. 따라서, 이 경우에는 마지막 논 제로 변환 계수의 스캐닝 위치에 대한 정보를 시그널링하는 것은 생략될 수 있다. 대신, 가장 먼저 코딩되는 첫번째 서브 블록은 변환 블록 내 좌상단 서브 블록일 수 있다. 한편, 상기 논 제로 변환 계수는 유효 계수(significant coefficient)라고 나타낼 수도 있다.

[244] 서브 블록 CBF: 마지막 논 제로 변환 계수의 스캐닝 위치에 대한 정보의 시그널링의 부재는 변환 스킵이 적용되고, coded_sub_block_flag를 갖는 서브 블록의 CBF 시그널링을 다음과 같이 수정해야한다.

[245] 양자화로 인해, 상술한 중요하지 않은 레벨의 시퀀스는 여전히 변환 블록

내에서 국부적으로 발생할 수 있다. 따라서, 마지막 논 제로 변환 계수의 스캐닝 위치에 대한 정보는 전술한 바와 같이 제거되고, coded_sub_block_flag는 모든 서브 블록에 대해 코딩될 수 있다.

[246] 또한, DC 주파수 위치에 대한 서브 블록(좌상단 서브 블록)에 대한 coded_sub_block_flag는 특별한 케이스를 나타낼 수 있다. 예를 들어, VVC Draft 3에서, 상기 좌상단 서브 블록에 대한 coded_sub_block_flag는 시그널링되지 않고 항상 1과 동일하다고 도출될 수 있다. 마지막 논 제로 변환 계수의 스캐닝 위치가 상기 좌상단 서브 블록이외의 서브 블록에 위치하는 경우는 DC 서브 블록(즉, 상기 좌상단 서브 블록) 외부에 적어도 하나의 유효 레벨(significant level)이 있음을 나타낼 수 있다. 결과적으로, 상기 DC 서브 블록에 대한 coded_sub_block_flag가 1 인 것으로 도출되지만 0/비유효(non-significant) 레벨만을 포함할 수 있다. 상술한 내용과 같이 현재 블록에 변환 스킵이 적용되고, 마지막 논 제로 변환 계수의 스캐닝 위치에 대한 정보가 없으면, 각 서브 블록에 대한 coded_sub_block_flag가 시그널링될 수 있다. 여기에는 상기 DC 서브 블록 이외의 모든 서브 블록들에 대한 coded_sub_block_flag 가 이미 0 인 경우를 제외하고 DC 서브 블록에 대한 coded_sub_block_flag도 포함될 수 있다. 한편, 예를 들어, 변환 블록의 스캐닝 순서로 우하단에서 좌상단으로 이동하는 대각 스캔 순서가 적용되고, DC 서브 블록에 대한 coded_sub_block_flag가 시그널링되지 않는 경우, 상기 DC 서브 블록에 대한 coded_sub_block_flag가 1과 같다고 도출될 수 있다(inferDcSbCbf = 1). 따라서, 상기 DC 서브 블록에는 적어도 하나의 유효 레벨이 있어야하므로 상기 DC 서브 블록 내 (0,0)의 첫번째 위치에 대한 sig_coeff_flag 이외의 sig_coeff_flag 들이 모두 0인 경우, 상기 (0,0)의 첫번째 위치에 대한 sig_coeff_flag 는 시그널링되지 않고, 1 과 동일하다고 도출될 수 있다(inferSbDcSigCoeffFlag = 1).

[247] 또한, coded_sub_block_flag의 컨텍스트 모델링이 변경될 수 있다. 예를 들어, 컨텍스트 모델 인덱스는 현재 서브 블록의 좌측 서브 블록의 coded_sub_block_flag 및 상기 현재 서브 블록의 상측 서브 블록의 coded_sub_block_flag의 합과 상기 coded_sub_block_flag 들의 논리적 분리로 계산될 수 있다.

[248] sig_coeff_flag 컨텍스트 모델링: sig_coeff_flag 컨텍스트 모델링의 로컬 템플릿(local template)는 현재 스캐닝 위치의 좌측 위치(NB0)와 상측 위치(NB1)만 포함하도록 수정될 수 있다. 컨텍스트 모델 오프셋은 유효 주변 위치의 sig_coeff_flag [NB0] + sig_coeff_flag [NB1]의 수로 도출될 수 있다. 따라서, 현재 변환 블록의 대각선 d에 따라 상이한 컨텍스트 세트의 선택이 제거될 수 있다. 그 결과 sig_coeff_flag 를 코딩하기 위해 3개의 컨텍스트 모델들과 단일 컨텍스트 모델이 설정될 수 있다.

[249] abs_level_gt1_flag 및 par_level_flag 컨텍스트 모델링: abs_level_gt1_flag 및 par_level_flag에는 단일 컨텍스트 모델이 사용될 수 있다. 또는,

abs_level_gt1_flag는 주변 coefficient의 non-zero coefficient의 개수에 의해 컨텍스트 모델이 결정될 수 있다.

[250] abs_remainder 코딩: 변환 스킵 레지듀얼 절대 레벨의 경험적 분포는 여전히 라플라시안 또는 기하 분포에 맞지만, 변환 계수 절대 레벨보다 더 큰 불안정성이 존재할 수 있다. 특히, 연속적인 실현의 윈도우 내의 분산은 레지듀얼 절대 레벨에 대해 더 높을 수 있다. 이에, abs_remainder 의 이진화 및 컨텍스트 모델링은 다음과 같이 수정될 수 있다.

[251] 예를 들어, abs_remainder 의 이진화에 더 높은 컷오프(cutoff) 값이 사용될 수 있다. 이를 통하여, sig_coeff_flag, abs_level_gt1_flag, par_level_flag 및 abs_level_gt3_flag를 사용한 코딩에서 abs_remainder에 대한 라이스 코드로의 전환 지점 및 각 빈 위치에 대한 전용 컨텍스트 모델에 더 높은 압축 효율이 제공될 수 있다. 상기 컷오프를 증가시키면 "X보다 큰" 플래그(예를 들어, abs_level_gt5_flag, abs_level_gt7_flag 등)가 컷오프에 도달할 때까지 더 많이 발생할 수 있다. 컷오프는 5로 고정될 수 있다(numGtFlags = 5).

[252] 또한, 라이스 파라미터 도출에 대한 템플릿이 수정될 수 있다. 즉, 현재 스캐닝 위치의 우측 주변 위치 및 하측 주변 위치만 sig_coeff_flag 컨텍스트 모델링의 로컬 템플릿으로 고려될 수 있다. 또는, 변환 스킵 적용 여부에 따라, 변환 스킵이 적용 되는 블록의 경우 라이스 파라미터는 1로 고정될 수 있다.

[253] coeff_sign_flag 컨텍스트 모델링: 부호(sign) 시퀀스 내부의 불안정성 및 예측 레지듀얼이 잔차가 종종 바이어스(bias)되는 것으로 인해, 글로벌 경험적 분포(distribution)가 거의 균일하게 분포된 경우에도 부호 관련 정보는 컨텍스트 모델을 사용하여 코딩될 수 있다. 싱글 전용 컨텍스트 모델이 부호 관련 정보의 코딩에 사용될 수 있고, 부호 관련 정보는 sig_coeff_flag 이후에 파싱되어 모든 컨텍스트 부호화 빈들(context coded bins)과 함께 유지될 수 있다.

[254] 컨텍스트 부호화 빈들의 감소: 첫 번째 스캐닝 패스에 대한 신텍스 엘리먼트들(syntax elements), 즉, sig_coeff_flag, abs_level_gt1_flag 및 par_level_flag의 전송은 변경되지 않을 수 있다. 그러나 샘플 당 컨텍스트 부호화 빈들(Context Coded Bins per sample, CCBs)의 최대치의 제한은 제거되고 다르게 조절될 수 있다. CCB 의 감소는 CCB> k 인 경우 유효하지 않은 모드로 지정하여 도출될 수 있다. 여기서, k는 양의 정수(positive integer)일 수 있다. 예를 들어, 정규 레벨 코딩 모드(regular level coding mode)의 경우, k = 2 일 수 있다. 상술한 제한은 양자화 공간의 감소에 대응할 수 있다.

[255] 한편, 상술한 레지듀얼 정보에 포함된 컨텍스트 기반으로 코딩된 신텍스 엘리먼트의 컨텍스트 모델을 가리키는 컨텍스트 인덱스(ctxIdx)는 후술하는 설명과 같이 도출될 수 있다.

[256] 예를 들어, 신텍스 엘리먼트에 대한 컨텍스트 인덱스를 도출하는 과정의 입력은 상기 신텍스 엘리먼트에 대한 빈 스트링에서 현재 빈의 위치를 나타내는 binIdx 가 일 수 있고, ctxTable, ctxIdx 및 bypassFlag 가 출력으로 도출될 수 있다.

[257] 먼저, 신텍스 엘리먼트에 대한 현재 빈에 대한 컨텍스트 인덱스 증분(context index increment, ctxInc)가 도출될 수 있다. 즉, 상기 신텍스 엘리먼트에 대한 현재 빈의 위치를 나타내는 binIdx 를 기반으로 ctxInc 가 도출될 수 있다. 상기 ctxInc 는 컨텍스트 증가 파라미터(context increment parameter)라고 나타낼 수도 있다.

[258] 신텍스 엘리먼트에 대한 binIdx 에 따라 도출되는 ctxInc 는 다음의 표와 같을 수 있다.

[259] [표16]

Syntax element	binIdx					
	0	1	2	3	4	>= 5
end_of_tile_group_flag	terminate	na	na	na	na	na
alf_ctb_flag[ ][ ][ ]	0..8 (clause 9.5.4.2.2)	na	na	na	na	na
sao_merge_left_flag	0	na	na	na	na	na
sao_merge_up_flag	0	na	na	na	na	na
sao_type_idx_luma	0	bypass	na	na	na	na
sao_type_idx_chroma	0	bypass	na	na	na	na
sao_offset_abs[ ][ ][ ][ ]	bypass	bypass	bypass	bypass	bypass	na
sao_offset_sign[ ][ ][ ][ ]	bypass	na	na	na	na	na
sao_band_position[ ][ ][ ]	bypass	bypass	bypass	bypass	bypass	bypass
sao_eo_class_luma	bypass	bypass	na	na	na	na
sao_eo_class_chroma	bypass	bypass	na	na	na	na
qt_split_cu_flag[ ][ ]	0..5 (clause 9.5.4.2.2)	na	na	na	na	na
mtt_split_cu_flag	0..12 (clause 9.5.4.2.2)	na	na	na	na	na
mtt_split_cu_vertical_flag	(cbWidth == cbHeight) ? 0 : ((cbWidth > cbHeight) ? 1 : 2)	na	na	na	na	na
mtt_split_cu_binary_flag	0	na	na	na	na	na
cu_skip_flag[ ][ ]	0,1,2 (clause 9.5.4.2.2)	na	na	na	na	na
pred_mode_flag	0	na	na	na	na	na
pcm_flag[ ][ ]	terminate	na	na	na	na	na
intra_luma_ref_idx[ ][ ]	0	1	2	na	na	na
intra_subpartitions_mode_flag	0	na	na	na	na	na
intra_subpartition_split_flag	0	na	na	na	na	na
intra_luma_mpm_flag[ ][ ]	0	na	na	na	na	na

[260]

intra_luma_mpm_idx[ ][ ]	bypass	bypass	bypass	bypass	bypass	na
intra_luma_mpm_remainder[ ][ ]	bypass	bypass	bypass	bypass	bypass	bypass
intra_chroma_pred_mode[ ][ ] sps_cclm_enabled_flag == 0	0	bypass	bypass	na	na	na
intra_chroma_pred_mode[ ][ ] sps_cclm_enabled_flag == 1 && bin at binIdx equal to 2 == 0	0	1	2	bypass	bypass	na
intra_chroma_pred_mode[ ][ ] sps_cclm_enabled_flag == 1 && bin at binIdx equal to 2 == 1	0	1	2	2	na	na
merge_subblock_flag[ ][ ]	0,1,2 (clause 9.5.4.2.2)	na	na	na	na	na
merge_subblock_idx[ ][ ] sps_sbtmvp_enabled_flag == 0	0	bypass	bypass	bypass	bypass	bypass
merge_subblock_idx[ ][ ] sps_sbtmvp_enabled_flag == 1	0	1	2	3	4	4
merge_flag[ ][ ]	0	na	na	na	na	na
mmvd_flag[ ][ ]	0	na	na	na	na	na

[261]

mmvd_merge_flag[ ][ ]	0	na	na	na	na	na
mmvd_distance_idx[ ][ ]	0	bypass	bypass	bypass	bypass	bypass
mmvd_direction_idx[ ][ ]	bypass	bypass	na	na	na	na
merge_triangle_flag[ ][ ]	0,1,2 (clause 9.5.4.2.2)	na	na	na	na	na
merge_triangle_idx[ ][ ]	0	bypass	bypass	bypass	bypass	bypass
merge_idx[ ][ ]	0	bypass	bypass	bypass	bypass	na
ciip_flag[ ][ ]	0	na	na	na	na	na
ciip_luma_mpm_flag[ ][ ]	0	na	na	na	na	na
ciip_luma_mpm_idx[ ][ ]	bypass	bypass	na	na	na	na
inter_pred_idc[x0][y0]	(cbWidth + cbHeight) != 8 ? 7 - ((1 + Log2(cbWidth) + Log2(cbHeight)) >> 1) : 4	4	na	na	na	na
inter_affine_flag[ ][ ]	0,1,2 (clause 9.5.4.2.2)	na	na	na	na	na
cu_affine_type_flag[ ][ ]	0	na	na	na	na	na
ref_idx_l0[ ][ ]	0	1	bypass	bypass	bypass	bypass
ref_idx_l1[ ][ ]	0	1	bypass	bypass	bypass	bypass
mvp_l0_flag[ ][ ]	0	na	na	na	na	na
mvp_l1_flag[ ][ ]	0	na	na	na	na	na
amvr_flag[ ][ ]	0,1,2 (clause 9.5.4.2.2)	na	na	na	na	na

[262]

amvr_4pel_flag[ ][ ]	0	na	na	na	na	na
gbi_idx[ ][ ] NoBackwardPredFlag == 0	0	1	na	na	na	na
gbi_idx[ ][ ] NoBackwardPredFlag == 1	0	1	2	3	na	na
cu_cbf	0	na	na	na	na	na
cu_sbt_flag	(cbWidth * cbHeight < 256) ? 1 : 0	na	na	na	na	na
cu_sbt_quad_flag	0	na	na	na	na	na
cu_sbt_horizontal_flag	(cbWidth == cbHeight) ? 0 : (cbWidth < cbHeight) ? 1 : 2	na	na	na	na	na
cu_sbt_pos_flag	0	na	na	na	na	na
abs_mvd_greater0_flag[ ]	0	na	na	na	na	na
abs_mvd_greater1_flag[ ]	0	na	na	na	na	na
abs_mvd_minus2[ ]	bypass	bypass	bypass	bypass	bypass	bypass
mvd_sign_flag[ ]	bypass	na	na	na	na	na

[263]

tu_cbf_luma[ ][ ][ ]	0,1,2,3 (clause 9.5.4.2.4)	na	na	na	na	na
tu_cbf_cb[ ][ ][ ]	trDepth = = 0 ? 0 : 1	na	na	na	na	na
tu_cbf_cr[ ][ ][ ]	tu_cbf_cb[ ][ ][ ]	na	na	na	na	na
cu_qp_delta_abs	0	1	1	1	1	bypass
cu_qp_delta_sign_flag	bypass	na	na	na	na	na
transform_skip_flag[ ][ ]	0	na	na	na	na	na
tu_mts_idx[ ][ ]	cqtDepth	6	7	8	na	na
last_sig_coeff_x_prefix	0..23 (clause 9.5.4.2.3)					
last_sig_coeff_y_prefix	0..23 (clause 9.5.4.2.3)					
last_sig_coeff_x_suffix	bypass	bypass	bypass	bypass	bypass	bypass
last_sig_coeff_y_suffix	bypass	bypass	bypass	bypass	bypass	bypass
coded_sub_block_flag[ ][ ]	0..3 (clause 9.5.4.2.5)	na	na	na	na	na
sig_coeff_flag[ ][ ]	0..89 (clause 9.5.4.2.7)	na	na	na	na	na
par_level_flag[ ]	0..32 (clause 9.5.4.2.8)	na	na	na	na	na
abs_level_gt1_flag[ ]	0..32 (clause 9.5.4.2.8)	na	na	na	na	na
abs_level_gt3_flag[ ]	0..32 (clause 9.5.4.2.8)	na	na	na	na	na
abs_remainder[ ]	bypass	bypass	bypass	bypass	bypass	bypass
dec_abs_level[ ]	bypass	bypass	bypass	bypass	bypass	bypass
coeff_sign_flag[ ]	bypass	na	na	na	na	na

[264] 표 16에서 "bypass", "terminate" 또는 "na"가 아닌 값을 갖는 빈의 컨텍스트 인덱스는 다음과 같이 도출될 수 있다.

[265] 신텍스 엘리먼트의 현재 빈에 대한 ctxInc 는 표 16에 상기 현재 빈에 대한 항목으로 지정된 값으로 도출될 수 있다. 또한, 상기 현재 빈에 대한 항목으로 지정된 값이 복수인 경우, 상기 항목에서 괄호 안에 개시된 절의 과정을 통하여 상기 ctxInc 가 도출될 수 있다. 상기 절은 VVC 표준에 개시된 절을 의미할 수 있다. 이후, 변수 ctxIdxOffset 가 initType의 현재 값에 따라 ctxIdx의 가장 낮은 값으로 지정될 수 있다. 상기 initType은 현재 블록을 포함하는 현재 슬라이스의 슬라이스 타입에 따라 결정될 수 있다. 예를 들어, 상기 슬라이스 타입이 I 슬라이스이면 initType은 0으로, 슬라이스 타입이 P 슬라이스이면 initType은 2로, 그 외의 경우 initType은 1로 될 수 있다. 또한, 신텍스 엘리먼트의 현재 빈에 대한 컨텍스트 인덱스(ctxIdx)는 ctxInc와 ctxIdxOffset의 합과 동일하게 설정될 수 있다. 즉, 현재 블록에 대한 상기 컨텍스트 인덱스는 ctxInc를 기반으로 결정될 수 있다. 또한, bypassFlag 는 0으로 설정될 수 있다.

[266] 한편, 표 16에서 상기 현재 빈에 대한 항목이 "bypass"인 경우, 상기 빈의 컨텍스트 인덱스는 다음과 같이 도출될 수 있다. 예를 들어, 상기 현재 빈에 대한 ctxTable 은 0으로 설정될 수 있다. 또한, 상기 현재 빈에 대한 컨텍스트 인덱스(ctxIdx)는 0으로 설정될 수 있다. 또한, bypassFlag 는 1로 설정될 수 있다.

[267] 한편, 표 16에서 상기 현재 빈에 대한 항목이 "terminate"인 경우, 상기 빈의 컨텍스트 인덱스는 다음과 같이 도출될 수 있다. 예를 들어, 상기 현재 빈에 대한 ctxTable 은 0으로 설정될 수 있다. 또한, 상기 현재 빈에 대한 컨텍스트 인덱스(ctxIdx)는 0으로 설정될 수 있다. 또한, bypassFlag 는 0으로 설정될 수

있다.

[268] 한편, 표 16에서 상기 현재 빈에 대한 항목이 "na"인 경우, 상기 빈에 대한 신텍스 엘리먼트들, 즉, 상기 빈에 대한 컨텍스트 인덱스는 다음과 같이 도출될 수 있다. 예를 들어, 상기 현재 빈에 대한 ctxIdx, ctxTable 및/또는 bypassFlag 는 발생하지 않을 수 있다.

[269] "bypass", "terminate" 또는 "na"가 아닌 값을 갖는 빈에 대한 ctxInc를 도출하기 위한 절들의 과정들은 후술하는 바와 같을 수 있다.

[270] 예를 들어, 9.5.4.2.2 절에 따른 ctxInc 를 도출하는 과정은 다음의 표와 같을 수 있다.

[271] [표17]

9.5.4.2.2 Derivation process of ctxInc using left and above syntax elements

Input to this process is the luma location (x0, y0) specifying the top-left luma sample of the current luma block relative to the top-left sample of the current picture, the colour component cIdx, the current coding quadtree depth cqDepth, and the width and the height of the current coding block in luma samples cbWidth and cbHeight.

Output of this process is ctxInc.

The location (xNbL, yNbL) is set equal to (x0 − 1, y0) and the variable availableL, specifying the availability of the block located directly to the left of the current block, is derived by invoking the availability derivation process for a block in z-scan order as specified in subclause 6.4 with the location (xCurr, yCurr) set equal to (x0, y0) and the neighbouring location (xNbY, yNbY) set equal to (xNbL, yNbL) as inputs, and the output is assigned to availableL.

The location (xNbA, yNbA) is set equal to (x0, y0 − 1) and the variable availableA specifying the availability of the coding block located directly above the current block, is derived by invoking the availability derivation process for a block in z-scan order as specified in subclause 6.4 with the location (xCurr, yCurr) set equal to (x0, y0) and the neighbouring location (xNbY, yNbY) set equal to (xNbA, yNbA) as inputs, and the output is assigned to availableA.

The variables sizeC, sizeTh2 and sizeTh1 are derived as follows:

sizeTh2 = (MaxBtSizeY = = 128) ? 1024 : ((MaxBtSizeY = = 64) ? 512 : 256) (9 19)

sizeTh1 = (MaxBtSizeY = = 128) ? 128 : 64 (9 20)

sizeC = cbWidth * cbHeight (9 21)

The assignment of ctxInc is specified as follows with condL and condA for the syntax elements alf_ctb_flag[x0][y0][cIdx], qt_split_cu_flag[x0][y0], mtt_split_cu_flag[x0][y0], cu_skip_flag[x0][y0], amvr_flag[x0][y0], inter_affine_flag[x0][y0], merge_triangle_flag[x0][y0] and merge_subblock_flag[x0][y0] specified in Table 9 11:

ctxInc = (condL && availableL) + (condA && availableA) + ctxSetIdx * 3 (9 22)

[272] 또한, 예를 들어, 9.5.4.2.3 절에 따른 ctxInc 를 도출하는 과정은 다음의 표와 같을 수 있다.

[273] [표18]

9.5.4.2.3 Derivation process of ctxInc for the syntax elements last_sig_coeff_x_prefix and last_sig_coeff_y_prefix
Inputs to this process are the variable binIdx, the colour component index cIdx, the binary logarithm of the transform block width log2TbWidth and the transform block height log2TbHeight.
Output of this process is the variable ctxInc.
The variable log2TbSize is derived as follows:
– If the syntax element to be parsed is last_sig_coeff_x_prefix, log2TbSize is set equal to log2TbWidth.
– Otherwise (the syntax element to be parsed is last_sig_coeff_y_prefix), log2TbSize is set equal to log2TbHeight.
The variables ctxOffset and ctxShift are derived as follows:
– If cIdx is equal to 0, ctxOffset is set equal to 3 * (log2TbSize − 2) + ((log2TbSize − 1) >> 2) and ctxShift is set equal to (log2TbSize + 1) >> 2.
– Otherwise (cIdx is greater than 0), ctxOffset is set equal to 21 and ctxShift is set equal to Clip3(0, 2, 2log2TbSize >> 3).
The variable ctxInc is derived as follows:
ctxInc = (binIdx >> ctxShift) + ctxOffset (9 23)

[274] 또한, 예를 들어, 9.5.4.2.4 절에 따른 ctxInc 를 도출하는 과정은 다음의 표와 같을 수 있다.

[275] [표19]

9.5.4.2.4 Derivation process of ctxInc for the syntax element tu_cbf_luma
Inputs to this process are the variable binIdx, the colour component index cIdx, the binary logarithm of the transform block width log2TbWidth and the transform block height log2TbHeight and the current transform tree depth trDepth.
Output of this process is the variable ctxInc.
The variable ctxInc is derived as follows:
– If IntraSubpartitionSplitType is equal to ISP_NO_SPLIT or cIdx is not equal to 0, the following applies:
ctxInc = trDepth = = 0 ? 1 : 0 (9 24)
– Otherwise (IntraSubpartitionSplitType is not equal to ISP_NO_SPLIT and cIdx is equal to 0), the following applies:
– The variable prevTuCbfY is derived as follows:
– If the current transform unit is the first one to be parsed in a coding unit, prevTuCbfY is set equal to 0.
– Otherwise, prevTuCbfY is set equal to the value of tu_cbf_luma of the previous luma transform unit in the current coding unit.
– The variable ctxInc is derived as follows:
ctxInc = 2 + prevTuCbfY (9 25)

[276] 또한, 예를 들어, 9.5.4.2.5 절에 따른 ctxInc 를 도출하는 과정은 다음의 표와 같을 수 있다.

[277] [표20]

9.5.4.2.5 Derivation process of ctxInc for the syntax element coded_sub_block_flag

Inputs to this process are the colour component index cIdx, the current sub-block scan location (xS, yS), the previously decoded bins of the syntax element coded_sub_block_flag and the binary logarithm of the transform block width log2TbWidth and the transform block height log2TbHeight.

Output of this process is the variable ctxInc.

The variable csbfCtx is derived using the current location (xS, yS), two previously decoded bins of the syntax element coded_sub_block_flag in scan order, log2TbWidth and log2TbHeight, as follows:

– The variables log2SbWidth and log2SbHeight are dervied as follows:

log2SbWidth = (Min(log2TbWidth, log2TbHeight) < 2 ? 1 : 2) (9 26)

log2SbHeight = log2SbWidth (9 27)

– The variables log2SbWidth and log2SbHeight are modifed as follows:

– If log2TbWidth is less than 2 and cIdx is equal to 0, the following applies

log2SbWidth = log2TbWidth (9 28)

log2SbHeight = 4 – log2SbWidth (9 29)

– Otherwise, if log2TbHeight is less than 2 and cIdx is equal to 0, the following applies

log2SbHeight = log2TbHeight (9 30)

log2SbWidth = 4 – log2SbHeight (9 31)

– csbfCtx is initialized with 0 as follows:

csbfCtx = 0 (9 32)

[278]

– When xS is less than (1 << (log2TbWidth – log2SbWidth)) – 1, csbfCtx is modified as follows:

csbfCtx += coded_sub_block_flag[xS + 1][yS] (9 33)

– When yS is less than (1 << (log2TbHeight – log2SbHeight)) – 1, csbfCtx is modified as follows:

csbfCtx += coded_sub_block_flag[xS][yS + 1] (9 34)

The context index increment ctxInc is derived using the colour component index cIdx and csbfCtx as follows:

– If cIdx is equal to 0, ctxInc is derived as follows:

ctxInc = Min(csbfCtx, 1) (9 35)

– Otherwise (cIdx is greater than 0), ctxInc is derived as follows:

ctxInc = 2 + Min(csbfCtx, 1) (9 36)

[279] 또한, 예를 들어, 9.5.4.2.6 절에 따른 ctxInc 를 도출하는 과정은 다음의 표와 같을 수 있다.

[280] [표21]

9.5.4.2.6 Derivation process for the variables locNumSig, locSumAbsPass1
Inputs to this process are the colour component index cIdx, the luma location (x0, y0) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture, the current coefficient scan location (xC, yC), the binary logarithm of the transform block width log2TbWidth, and the binary logarithm of the transform block height log2TbHeight.
Outputs of this process are the variables locNumSig and locSumAbsPass1.
Given the syntax elements sig_coeff_flag[x][y] and the array AbsLevelPass1[x][C] for the transform block with component index cIdx and the top-left luma location (x0, y0), the variables locNumSig and locSumAbsPass1 are derived as specified by the following pseudo code:

```
locNumSig      = 0
locSumAbsPass1 = 0
if( xC < (1 << log2TbWidth) − 1 ) {
  locNumSig      += sig_coeff_flag[ xC + 1 ][ yC ]
  locSumAbsPass1 += AbsLevelPass1[ xC + 1 ][ yC ]
  if( xC < (1 << log2TbWidth) − 2 ) {
    locNumSig      += sig_coeff_flag[ xC + 2 ][ yC ]
    locSumAbsPass1 += AbsLevelPass1[ xC + 2 ][ yC ]
  }
  if( yC < (1 << log2TbHeight) − 1 ) {
    locNumSig      += sig_coeff_flag[ xC + 1 ][ yC + 1 ]      (9 37)
    locSumAbsPass1 += AbsLevelPass1[ xC + 1 ][ yC + 1 ]
  }
}
if( yC < (1 << log2TbHeight) − 1 ) {
  locNumSig      += sig_coeff_flag[ xC ][ yC + 1 ]
  locSumAbsPass1 += AbsLevelPass1[ xC ][ yC + 1 ]
  if( yC < (1 << log2TbHeight) − 2 ) {
    locNumSig      += sig_coeff_flag[ xC ][ yC + 2 ]
    locSumAbsPass1 += AbsLevelPass1[ xC ][ yC + 2 ]
  }
}
```

[281] 또한, 예를 들어, 9.5.4.2.7 절에 따른 ctxInc 를 도출하는 과정은 다음의 표와 같을 수 있다.

[282] [표22]

9.5.4.2.7 Derivation process of ctxInc for the syntax element sig_coeff_flag
Inputs to this process are the colour component index cIdx, the luma location (x0, y0) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture, the current coefficient scan location (xC, yC), the binary logarithm of the transform block width log2TbWidth, and the binary logarithm of the transform block height log2TbHeight.
Output of this process is the variable ctxInc.
The variable locSumAbsPass1 is derived by invoking the derivation process for the variables locNumSig and locSumAbsPass1 specifies in clause 9.5.4.2.6 with colour component index cIdx, the luma location (x0, y0), the current coefficient scan location (xC, yC), the binary logarithm of the transform block width log2TbWidth, and the binary logarithm of the transform block height log2TbHeight as input.
The variable d is set equal to xC + yC.
The variable ctxInc is derived as follows:
– If cIdx is equal to 0, ctxInc is derived as follows:
ctxInc = 18 * Max(0, QState − 1) + Min(locSumAbsPass1, 5) + (d < 2 ? 12 : (d < 5 ? 6 : 0)) (9 38)
– Otherwise (cIdx is greater than 0), ctxInc is derived as follows:
ctxInc = 54 + 12 * Max(0, QState − 1) + Min(locSumAbsPass1, 5) + (d < 2 ? 6 : 0) (9 39)

[283] 또한, 예를 들어, 9.5.4.2.8 절에 따른 ctxInc 를 도출하는 과정은 다음의 표와 같을 수 있다.

[284] [표23]

9.5.4.2.8 Derivation process of ctxInc for the syntax elements par_level_flag, abs_level_gt1_flag, and abs_level_gt3_flag
Inputs to this process are the colour component index cIdx, the luma location (x0, y0) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture, the current coefficient scan location (xC, yC), the binary logarithm of the transform block width log2TbWidth, and the binary logarithm of the transform block height log2TbHeight.
Output of this process is the variable ctxInc.
The variablea locNumSig and locSumAbsPass1 is derived by invoking the derivation process for the variables locNumSig and locSumAbsPass1 specifies in clause 9.5.4.2.6 with colour component index cIdx, the luma location (x0, y0), the current coefficient scan location (xC, yC), the binary logarithm of the transform block width log2TbWidth, and the binary logarithm of the transform block height log2TbHeight as input.
The variable ctxOffset is set equal to Min(locSumAbsPass1 − locNumSig, 4).
The variable d is set equal to xC + yC.
The variable ctxInc is derived as follows:
– If xC is equal to LastSignificantCoeffX and yC is equal to LastSignificantCoeffY, ctxInc is derived as follows:
ctxInc = (cIdx == 0 ? 0 : 21) (9 40)
– Otherwise, if cIdx is equal to 0, ctxInc is derived as follows:
ctxInc = 1 + ctxOffset + (d == 0 ? 15 : (d < 3 ? 10 : (d < 10 ? 5 : 0))) (9 41)
– Otherwise (cIdx is greater than 0), ctxInc is derived as follows:
ctxInc = 22 + ctxOffset + (d == 0 ? 5 : 0) (9 42)

[285] 한편, 상술한 내용과 같이 변환 인코딩을 수행하지 않는 블록, 즉, 변환이 적용되지 않은 레지듀얼 계수들을 포함하는 변환 블록은 일반적인 변환

인코딩이 수행된 블록과 레지듀얼 정보의 특성이 다르므로, 변환 인코딩을 수행하지 않는 블록 위한 효율적인 잔여 데이터 부호화 방법이 필요하다. 한편, 상술한 내용과 같이 변환 적용 여부를 나타내는 변환 스킵 플래그는 변환 블록 단위로 전송될 수 있으며, 본 문서에서는 변환 블록의 사이즈를 한정하지 않는다. 예를 들어, 변환 스킵 플래그의 값이 1인 경우, 본 문서에서 제안하는 레지듀얼 정보 인코딩/디코딩 방안이 수행될 수 있으며, 상기 변환 스킵 플래그의 값이 0인 경우, 상술한 표 4에서 설명한 기존의 레지듀얼 정보 인코딩/디코딩 방안이 수행될 수 있다. 또는, 상기 변환 스킵 플래그가 현재 블록에 변환이 적용되지 않음(변환이 스킵됨)을 나타내는 경우, 상술한 표 5에 개시된 변환 스킵 모드에서의 레지듀얼 정보 인코딩/디코딩 방안이 수행될 수도 있다.

[286] 여기서, 현재 블록의 변환 스킵 플래그는 현재 블록의 예측 모드와 상관성(correlation)이 존재할 수 있다. 또는, 현재 블록의 변환 스킵 플래그는 현재 블록의 좌측 또는 상측의 주변 블록들의 예측 모드와 상관성이 존재할 수도 있다. 예를 들어, 화면 내 블록 복사(intra block copy, IBC) 예측 모드가 사용되는 경우, 변환 스킵 블록의 발생 빈도가 증가할 수 있다.

[287] 따라서, 본 명세서의 일 실시예에서는 변환 스킵 플래그에 대한 컨텍스트 모델을 현재 블록의 예측 모드 정보 또는 현재 블록의 주변 블록들 중 적어도 하나에 기초하여 결정하고, 결정된 컨텍스트 모델을 기반으로 변환 스킵 플래그를 코딩하는 방안을 제안한다.

[288] 현재 블록의 예측 모드 정보에 기초하여 변환 스킵 플래그에 대한 컨텍스트 모델을 도출하는 일 실시예는 다음의 표 24와 같이 정의될 수 있다.

[289] [표24]

Syntax element	binIdx					
	0	1	2	3	4	>= 5
transform_skip_flag[ ][ ]	CuPredMode[ ][ ] != MODE_IBC ? 0 : 1	na	na	na	na	na

[290] 상술한 표 24을 참조하면 현재 블록에 변환 스킵이 적용되었는지 여부를 나타내는 신텍스 엘리먼트 변환 스킵 플래그(transform_skip_flag)의 빈에 대한 컨텍스트 모델은 현재 블록의 예측 모드(CuPredMode)를 기반으로 결정될 수 있다.

[291] 변환 스킵 플래그의 빈에 대한 컨텍스트 모델은 상기 변환 스킵 플래그에 대한 컨텍스트 인덱스 증분(context index increment, ctxInc)을 기반으로 결정될 수 있다.

[292] 예를 들어, 상기 변환 스킵 플래그에 대한 ctxInc는 현재 블록의 예측 모드가 IBC(Intra block copy) 예측 모드가 아닌 경우에는 '0'으로, 현재 블록의 예측 모드가 IBC 예측 모드인 경우에는 '1'로 도출될 수 있다.

[293] 한편, 상술한 내용과 같이 현재 레지듀얼 계수의 변환 스킵 플래그에 대한 상기 ctxInc이 도출된 이후에 상기 ctxInc을 기반으로 컨텍스트 모델을 도출하는 과정은 하기와 같을 수 있다.

[294] 예를 들어, 상기 변환 스킵 플래그에 대한 ctxIdxOffset 는 0, 2, 4 값 중 하나로 도출될 수 있다. 예를 들어, 상기 현재 슬라이스의 슬라이스 타입이 I 슬라이스인 경우 상기 ctxIdxOffset은 0으로 도출될 수 있고, 상기 현재 슬라이스의 슬라이스 타입이 P 슬라이스인 경우 상기 ctxIdxOffset은 4로 도출될 수 있고, 그 외의 경우 상기 ctxIdxOffset은 2로 도출될 수 있다. 다만 이는 예시로서 ctxIdxOffset은 그 외의 다른 값으로 미리 결정될 수도 있다. 따라서, 동일 슬라이스 내의 블록들에 대하여 상기 변환 스킵 플래그에 대한 컨텍스트 모델(또는 ctxIdx)은 ctxInc를 기반으로 결정될 수 있다.

[295] 예를 들어, 상술한 내용에서 ctxIdx (컨텍스트 모델을 나타내는 컨텍스트 모델 인덱스)는 ctxInc 0 또는 1 을 기반으로 도출될 수 있고, 이에 따라 제 1 컨텍스트 모델 내지 제2 컨텍스트 모델 중 하나가 도출될 수 있다. 또는, 예를 들어, 상기 변환 스킵 플래그에 대한 ctxIdx (컨텍스트 모델을 나타내는 컨텍스트 모델 인덱스)는 ctxInc 0 내지 ctxInc N 중 하나의 값을 기반으로 도출될 수 있고, 이에 따라 제 1 컨텍스트 모델 내지 제 N+1 컨텍스트 모델 중 하나가 도출될 수 있다.

[296] 따라서, 현재 블록의 예측 모드가 IBC 예측 모드가 아닌 경우 변환 스킵 플래그에 대한 ctxIdc 는 0으로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 1 ctxIdx로 도출(즉, 제 1 컨텍스트 모델로 도출)될 수 있다. 또한, 현재 블록의 예측 모드가 IBC 예측 모드인 경우 변환 스킵 플래그에 대한 ctxIdc 는 1로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제2 ctxIdx로 도출(즉, 제2 컨텍스트 모델로 도출)될 수 있다.

[297] 또는, 현재 블록의 예측 모드 정보에 기초하여 변환 스킵 플래그에 대한 컨텍스트 모델을 도출하는 다른 실시예는 다음의 표 25와 같이 정의될 수 있다.

[298] [표25]

Syntax element	binIdx					
	0	1	2	3	4	>= 5
transform_skip_flag[ ][ ]	CuPredMode[ ][ ] != MODE_INTER ? 0 : 1	na	na	na	na	na

[299] 상술한 표 25를 참조하면, 상기 변환 스킵 플래그에 대한 상기 ctxInc는 현재 블록의 예측 모드가 인터 예측 모드가 아닌 경우에는 '0'으로, 현재 블록의 예측 모드가 인터 예측 모드인 경우에는 1'로 도출될 수 있다.

[300] 따라서, 현재 블록의 예측 모드가 인터 예측 모드가 아닌 경우 변환 스킵 플래그에 대한 ctxIdc 는 0으로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 1 ctxIdx로 도출(즉, 제 1 컨텍스트 모델로 도출)될 수 있다. 또한, 현재 블록의 예측 모드가 인터 예측 모드인 경우 변환 스킵 플래그에 대한 ctxIdc 는 1로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제2 ctxIdx로

도출(즉, 제2 컨텍스트 모델로 도출)될 수 있다.

[301] 현재 블록의 예측 모드 정보에 기초하여 변환 스킵 플래그에 대한 컨텍스트 모델을 도출하는 또 다른 실시예는 다음의 표 26과 같이 정의될 수 있다.

[302] [표26]

Syntax element	binIdx					
	0	1	2	3	4	>= 5
transform_skip_flag[ ][ ]	CuPredMode[ ][ ] != MODE_INTRA ? 0 : 1	na	na	na	na	na

[303] 상술한 표 26을 참조하면, 상기 변환 스킵 플래그에 대한 상기 ctxInc는 현재 블록의 예측 모드가 인트라 예측 모드가 아닌 경우에는 '0'으로, 현재 블록의 예측 모드가 인트라 예측 모드인 경우에는 1'로 도출될 수 있다.

[304] 따라서, 현재 블록의 예측 모드가 인트라 예측 모드가 아닌 경우 변환 스킵 플래그에 대한 ctxIdc 는 0으로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 1 ctxIdx로 도출(즉, 제 1 컨텍스트 모델로 도출)될 수 있다. 또한, 현재 블록의 예측 모드가 인트라 예측 모드인 경우 변환 스킵 플래그에 대한 ctxIdc 는 1로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제2 ctxIdx로 도출(즉, 제2 컨텍스트 모델로 도출)될 수 있다.

[305] 한편, 현재 블록의 상측 주변 블록의 예측 모드 정보 및 좌측 주변 블록의 예측 모드 정보에 기초하여 변환 스킵 플래그에 대한 컨텍스트 모델을 도출하는 경우, 예를 들어 상기 ctxInc는 하기 표 27과 같이 0, 1 및 2 중 어느 하나로 도출될 수 있다.

[306] [표27]

Syntax element	binIdx					
	0	1	2	3	4	>= 5
transform_skip_flag[ ][ ]	0,1,2	na	na	na	na	na

[307] 이 때, 상기 ctxInc를 도출하는 일 실시예는 다음의 표 28 및 표 29와 같이 정의될 수 있다. 표 28 및 표 29에 따르면, 현재 블록의 상측 주변 블록 의 예측 모드 및 좌측 주변 블록의 예측 모드 각각이 IBC 예측 모드인지 여부에 따라 상기 ctxInc가 도출될 수 있다.

[308] [표28]

Input to this process is the luma location (x0, y0) specifying the top-left luma sample of the current luma block relative to the top-left sample of the current picture, the colour component cIdx, the current coding quadtree depth cqDepth, the width and the height of the current coding block in luma samples cbWidth and cbHeight, and the variables allowSplitBtVer, allowSplitBtHor, allowSplitTtVer, allowSplitTtHor, and allowSplitQt as derived in the coding tree semantics in clause 7.4.7.4.

Output of this process is ctxInc.

The location (xNbL, yNbL) is set equal to (x0 − 1, y0) and the variable availableL, specifying the availability of the block located directly to the left of the current block, is derived by invoking the availability derivation process for a block in z-scan order as specified in subclause 6.4 with the location (xCurr, yCurr) set equal to (x0, y0) and the neighbouring location (xNbY, yNbY) set equal to (xNbL, yNbL) as inputs, and the output is assigned to availableL.

The location (xNbA, yNbA) is set equal to (x0, y0 − 1) and the variable availableA specifying the availability of the coding block located directly above the current block, is derived by invoking the availability derivation process for a block in z-scan order as specified in subclause 6.4 with the location (xCurr, yCurr) set equal to (x0, y0) and the neighbouring location (xNbY, yNbY) set equal to (xNbA, yNbA) as inputs, and the output is assigned to availableA.

The assignment of ctxInc is specified as follows with condL and condA specified in For the syntax element pred mode flag[x0][y0]:

ctxInc = (condL && availableL) | | (condA && availableA) (28-1)

Table 35:

For the syntax elements alf_ctb_flag[x0][y0][cIdx], split_qt_flag, split_cu_flag, cu_skip_flag[x0][y0], pred_mode_ibc_flag[x0][y0], amvr_flag[x0][y0], inter_affine_flag[x0][y0], merge triangle flag[x0][y0] and merge subblock flag[x0][y0], tramsform skip flag [x0][y0] :

ctxInc = (condL && availableL) + (condA && availableA) + ctxSetIdx * 3 (28-2)

For the syntax element pred_mode_flag[x0][y0]:

ctxInc = (condL && availableL) | | (condA && availableA) (28-3)

[309] [표29]

Syntax element	condL	condA	ctxSetIdx
alf_ctb_flag[x0][y0][cIdx]	alf_ctb_flag[xNbL][yNbL][cIdx]	alf_ctb_flag[xNbA][yNbA][cIdx]	cIdx
qsplit_qt_flag	cqtDepth[xNbL][yNbL] > cqtDepth	cqtDepth[xNbA][yNbA] > cqtDepth	(cqtDepth < 2) ? 0 : 1
msplit_cu_flag	CbHeight[xNbL][yNbL] < cbHeight	CbWidth[xNbA][yNbA] < cbWidth	(allowSplitBtVer + allowSplitBtHor + allowSplitTtVer + allowSplitTtHor + 2 * allowSplitQt − 1) / 3
cu_skip_flag[x0][y0]	cu_skip_flag[xNbL][yNbL]	cu_skip_flag[xNbA][yNbA]	0
pred_mode_flag[x0][y0]	CuPredMode[xNbL][yNbL] = = MODE_INTRA	CuPredMode[xNbA][yNbA] = = MODE_INTRA	0
pred_mode_ibc_flag[x0][y0]	CuPredMode[xNbL][yNbL] = = MODE_IBC	CuPredMode[xNbA][yNbA] = = MODE_IBC	0
amvr_flag[x0][y0]	amvr_flag[xNbL][yNbL]	amvr_flag[xNbA][yNbA]	0
merge_subblock_flag[x0][y0]	merge_subblock_flag[xNbL][yNbL] | inter_affine_flag[xNbL][yNbL]	merge_subblock_flag[xNbA][yNbA] | inter_affine_flag[xNbA][yNbA]	0
merge_triangle_flag[x0][y0]	merge_triangle_flag [xNbL][yNbL]	merge_triangle_flag [xNbA][yNbA]	0
inter_affine_flag [x0][y0]	merge_subblock_flag[xNbL][yNbL] | inter_affine_flag[xNbL][yNbL]	merge_subblock_flag[xNbA][yNbA] | inter_affine_flag[xNbA][yNbA]	0
tramsform_skip_flag [x0][y0]	CuPredMode[xNbL][yNbL] = = MODE_IBC	CuPredMode[xNbA][yNbA] = = MODE_IBC	0

[310] 상술한 표 28의 수학식 (28-2)및 표 29를 참조하면, 상기 ctxInc는 현재 블록의

상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드가 모두 IBC(Intra block copy) 예측 모드가 아닌 경우에는'0'으로, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드 중 어느 하나가 IBC 예측 모드인 경우에는 '1'로 도출될 수 있고, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드가 모두 IBC 예측 모드인 경우에는 '2'로 도출될 수 있다.

[311] 따라서, 현재 블록의 상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드가 모두 IBC 예측 모드가 아닌 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 0으로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 1 ctxIdx로 도출(즉, 제 1 컨텍스트 모델로 도출)될 수 있다. 또한, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드 중 어느 하나가 IBC 예측 모드인 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 1로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 2 ctxIdx로 도출(즉, 제 2 컨텍스트 모델로 도출)될 수 있다. 또한, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드가 모두 IBC 예측 모드인 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 2로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 3 ctxIdx로 도출(즉, 제 3 컨텍스트 모델로 도출)될 수 있다.

[312] 또한, 상기 ctxInc를 도출하는 다른 실시예는 상기 표 28 및 하기 표 30과 같이 정의될 수 있다. 표 28 및 표 30에 따르면, 현재 블록의 상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드 각각이 인터 예측 모드인지 여부에 따라 상기 ctxInc가 도출될 수 있다.

[313] [표30]

Syntax element	condL	condA	ctxSetIdx
alf_ctb_flag[x0][y0][cIdx]	alf_ctb_flag[xNbL][yNbL][cIdx]	alf_ctb_flag[xNbA][yNbA][cIdx]	cIdx
qsplit_qt_flag	cqtDepth[xNbL][yNbL] > cqtDepth	cqtDepth[xNbA][yNbA] > cqtDepth	(cqtDepth < 2) ? 0 : 1
msplit_cu_flag	CbHeight[xNbL][yNbL] < cbHeight	CbWidth[xNbA][yNbA] < cbWidth	(allowSplitBtVer + allowSplitBtHor + allowSplitTtVer + allowSplitTtHor + 2 * allowSplitQt − 1) / 3
cu_skip_flag[x0][y0]	cu_skip_flag[xNbL][yNbL]	cu_skip_flag[xNbA][yNbA]	0
pred_mode_flag[x0][y0]	CuPredMode[xNbL][yNbL] = = MODE_INTRA	CuPredMode[xNbA][yNbA] = = MODE_INTRA	0
pred_mode_ibc_flag[x0][y0]	CuPredMode[xNbL][yNbL] = = MODE_IBC	CuPredMode[xNbA][yNbA] = = MODE_IBC	0
amvr_flag[x0][y0]	amvr_flag[xNbL][yNbL]	amvr_flag[xNbA][yNbA]	0
merge_subblock_flag[x0][y0]	merge_subblock_flag[xNbL][yNbL] | inter_affine_flag[xNbL][yNbL]	merge_subblock_flag[xNbA][yNbA] | inter_affine_flag[xNbA][yNbA]	0
merge_triangle_flag[x0][y0]	merge_triangle_flag [xNbL][yNbL]	merge_triangle_flag [xNbA][yNbA]	0
inter_affine_flag [x0][y0]	merge_subblock_flag[xNbL][yNbL] | inter_affine_flag[xNbL][yNbL]	merge_subblock_flag[xNbA][yNbA] | inter_affine_flag[xNbA][yNbA]	0
transform_skip_flag [x0][y0]	CuPredMode[xNbL][yNbL] = = MODE_INTER	CuPredMode[xNbA][yNbA] = = MODE_INTER	0

[314] 상술한 표 28의 수학식 (28-2)및 표 30을 참조하면, 상기 ctxInc는 현재 블록의 상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드가 모두 인터 예측 모드가 아닌 경우에는'0'으로, 상기 상측 주변 블록의 예측 모드 및 상기 좌측

주변 블록의 예측 모드 중 어느 하나가 인터 예측 모드인 경우에는 '1'로 도출될 수 있고, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드가 모두 인터 예측 모드인 경우에는 '2'로 도출될 수 있다.

[315] 따라서, 현재 블록의 상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드가 모두 인터 예측 모드가 아닌 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 0으로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 1 ctxIdx로 도출(즉, 제 1 컨텍스트 모델로 도출)될 수 있다. 또한, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드 중 어느 하나가 인터 예측 모드인 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 1로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 2 ctxIdx로 도출(즉, 제 2 컨텍스트 모델로 도출)될 수 있다. 또한, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드가 모두 인터 예측 모드인 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 2로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 3 ctxIdx로 도출(즉, 제 3 컨텍스트 모델로 도출)될 수 있다.

[316] 또한, 상기 ctxInc를 도출하는 다른 실시예는 상기 표 28 및 하기 표 31과 같이 정의될 수 있다. 표 28 및 표 31에 따르면, 현재 블록의 상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드 각각이 인트라 예측 모드인지 여부에 따라 상기 ctxInc가 도출될 수 있다.

[317] [표31]

Syntax element	condL	condA	ctxSetIdx
alf_ctb_flag[x0][y0][cIdx]	alf_ctb_flag[xNbL][yNbL][cIdx]	alf_ctb_flag[xNbA][yNbA][cIdx]	cIdx
qsplit_qt_flag	cqtDepth[xNbL][yNbL] > cqtDepth	cqtDepth[xNbA][yNbA] > cqtDepth	(cqtDepth < 2) ? 0 : 1
msplit_cu_flag	CbHeight[xNbL][yNbL] < cbHeight	CbWidth[xNbA][yNbA] < cbWidth	(allowSplitBtVer + allowSplitBtHor + allowSplitTtVer + allowSplitTtHor + 2 * allowSplitQt − 1) / 3
cu_skip_flag[x0][y0]	cu_skip_flag[xNbL][yNbL]	cu_skip_flag[xNbA][yNbA]	0
pred_mode_flag[x0][y0]	CuPredMode[xNbL][yNbL] = = MODE_INTRA	CuPredMode[xNbA][yNbA] = = MODE_INTRA	0
pred_mode_ibc_flag[x0][y0]	CuPredMode[xNbL][yNbL] = = MODE_IBC	CuPredMode[xNbA][yNbA] = = MODE_IBC	0
amvr_flag[x0][y0]	amvr_flag[xNbL][yNbL]	amvr_flag[xNbA][yNbA]	0
merge_subblock_flag[x0][y0]	merge_subblock_flag[xNbL][yNbL] | inter_affine_flag[xNbL][yNbL]	merge_subblock_flag[xNbA][yNbA] | inter_affine_flag[xNbA][yNbA]	0
merge_triangle_flag[x0][y0]	merge_triangle_flag [xNbL][yNbL]	merge_triangle_flag [xNbA][yNbA]	0
inter_affine_flag [x0][y0]	merge_subblock_flag[xNbL][yNbL] | inter_affine_flag[xNbL][yNbL]	merge_subblock_flag[xNbA][yNbA] | inter_affine_flag[xNbA][yNbA]	0
transform_skip_flag [x0][y0]	CuPredMode[xNbL][yNbL] = = MODE_INTRA	CuPredMode[xNbA][yNbA] = = MODE_INTRA	0

[318] 상술한 표 28의 수학식 (28-2)및 표 31을 참조하면, 상기 ctxInc는 현재 블록의 상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드가 모두 인트라 예측 모드가 아닌 경우에는'0'으로, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드 중 어느 하나가 인트라 예측 모드인 경우에는 '1'로 도출될 수 있고, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드가

모두 인트라 예측 모드인 경우에는 '2'로 도출될 수 있다.

[319] 따라서, 현재 블록의 상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드가 모두 인트라 예측 모드가 아닌 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 0으로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 1 ctxIdx로 도출(즉, 제 1 컨텍스트 모델로 도출)될 수 있다. 또한, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드 중 어느 하나가 인트라 예측 모드인 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 1로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 2 ctxIdx로 도출(즉, 제 2 컨텍스트 모델로 도출)될 수 있다. 또한, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드가 모두 인트라 예측 모드인 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 2로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 3 ctxIdx로 도출(즉, 제 3 컨텍스트 모델로 도출)될 수 있다.

[320] 한편, 현재 블록의 상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드 정보에 기초하여 변환 스킵 플래그에 대한 컨텍스트 모델을 도출하는 경우, 상기 ctxInc는 상술한 표 27과 달리 0 및 1 중 어느 하나로 도출될 수도 있으며, 이는 하기 표 32와 같다.

[321] [표32]

Syntax element	binIdx					
	0	1	2	3	4	>= 5
transform_skip_flag[ ][ ]	0,1	na	na	na	na	na

[322] 이 때, 상기 ctxInc를 도출하는 일 실시예는 다음의 표 33 및 상기 표 29와 같이 정의될 수 있다. 표 33 및 표 29에 따르면, 현재 블록의 상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드 각각이 IBC 예측 모드인지 여부에 따라 상기 ctxInc가 도출될 수 있다.

[323] [표33]

Input to this process is the luma location (x0, y0) specifying the top-left luma sample of the current luma block relative to the top-left sample of the current picture, the colour component cIdx, the current co ding quadtree depth cqDepth, the width and the height of the current coding block in luma samples cbWidth and cbHeight, and the variables allowSplitBtVer, allowSplitBtHor, allowSplitTtVer, allowSpli tTtHor, and allowSplitQt as derived in the coding tree semantics in clause 7.4.7.4.

Output of this process is ctxInc.

The location (xNbL, yNbL) is set equal to (x0 − 1, y0) and the variable availableL, specifying the availability of the block located directly to the left of the current block, is derived by invoking the availability derivation process for a block in z-scan order as specified in subclause 6.4 with the location (xCurr, yCurr) set equal to (x0, y0) and the neighbouring location (xNbY, yNbY) set equal to (xNbL, yNbL) as inputs, and the output is assigned to availableL.

The location (xNbA, yNbA) is set equal to (x0, y0 − 1) and the variable availableA specifying the availability of the coding block located directly above the current block, is derived by invoking the availability derivation process for a block in z-scan order as specified in subclause 6.4 with the location (xCurr, yCurr) set equal to (x0, y0) and the neighbouring location (xNbY, yNbY) set equal to (xNbA, yNbA) as inputs, and the output is assigned to availableA.

The assignment of ctxInc is specified as follows with condL and condA specified in For the syntax element pred mode flag[x0][y0]:

ctxInc = (condL && availableL) | | (condA && availableA) (33-1)

Table 9-16:

For the syntax elements alf_ctb_flag[x0][y0][cIdx], split_qt_flag, split_cu_flag, cu_skip_flag[x0][y0], pred_mode_ibc_flag[x0][y0], amvr_flag[x0][y0], inter_affine_flag[x0][y0], merge triangle flag[x0][y0] and merge subblock flag[x0][y0]:

ctxInc = (condL && availableL) + (condA && availableA) + ctxSetIdx * 3 (33-2)

For the syntax element pred_mode_flag[x0][y0] , tramsform_skip_flag [x0][y0]:

ctxInc = (condL && availableL) | | (condA && availableA) (33-3)

[324] 상술한 표 33의 수학식 (33-3) 및 표 29를 참조하면, 상기 ctxInc는 현재 블록의 상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드가 모두 IBC(Intra block copy) 예측 모드가 아닌 경우에는'0'으로, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드 중 적어도 하나가 IBC 예측 모드인 경우에는 '1'로 도출될 수 있다.

[325] 따라서, 현재 블록의 상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드가 모두 IBC 예측 모드가 아닌 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 0으로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 1 ctxIdx로 도출(즉, 제 1 컨텍스트 모델로 도출)될 수 있다. 또한, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드 중 적어도 하나가 IBC 예측 모드인 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 1로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 2 ctxIdx로 도출(즉, 제 2 컨텍스트 모델로 도출)될 수 있다.

[326] 또한, 상기 ctxInc를 도출하는 다른 실시예는 상기 표 33 및 표 30과 같이 정의될 수 있다. 표 33 및 표 30에 따르면, 현재 블록의 상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드 각각이 인터 예측 모드인지 여부에 따라 상기

ctxInc가 도출될 수 있다.

[327] 상술한 표 33의 수학식 (33-3) 및 표 30을 참조하면, 상기 ctxInc는 현재 블록의 상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드가 모두 인터 예측 모드가 아닌 경우에는'0'으로, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드 중 적어도 하나가 인터 예측 모드인 경우에는 '1'로 도출될 수 있다.

[328] 따라서, 현재 블록의 상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드가 모두 인터 예측 모드가 아닌 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 0으로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 1 ctxIdx로 도출(즉, 제 1 컨텍스트 모델로 도출)될 수 있다. 또한, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드 중 적어도 하나가 인터 예측 모드인 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 1로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 2 ctxIdx로 도출(즉, 제 2 컨텍스트 모델로 도출)될 수 있다.

[329] 상기 ctxInc를 도출하는 또 다른 실시예는 상기 표 33 및 표 31과 같이 정의될 수 있다. 표 33 및 표 31에 따르면, 현재 블록의 상측 주변 블록 및 좌측 주변 블록의 예측 모드 각각이 인트라 예측 모드인지 여부에 따라 상기 ctxInc가 도출될 수 있다.

[330] 상술한 표 33의 수학식 (33-3) 및 표 31을 참조하면, 상기 ctxInc는 현재 블록의 상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드가 모두 인트라 예측 모드가 아닌 경우에는'0'으로, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드 중 적어도 하나가 인트라 예측 모드인 경우에는 '1'로 도출될 수 있다.

[331] 따라서, 현재 블록의 상측 주변 블록의 예측 모드 및 좌측 주변 블록의 예측 모드가 모두 인트라 예측 모드가 아닌 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 0으로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 1 ctxIdx로 도출(즉, 제 1 컨텍스트 모델로 도출)될 수 있다. 또한, 상기 상측 주변 블록의 예측 모드 및 상기 좌측 주변 블록의 예측 모드 중 적어도 하나가 인트라 예측 모드인 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 1로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 2 ctxIdx로 도출(즉, 제 2 컨텍스트 모델로 도출)될 수 있다.

[332] 한편, 현재 블록의 변환 스킵 플래그에 대한 컨텍스트 모델은 현재 블록의 상측 주변 블록의 변환 스킵 플래그 정보 및 좌측 주변 블록 각각의 변환 스킵 플래그 정보에 기초하여 도출될 수도 있다.

[333] 따라서, 본 명세서의 일 실시예에서는 변환 스킵 플래그에 대한 컨텍스트 모델을 상측 주변 블록의 변환 스킵 플래그 정보 및 좌측 주변 블록들의 변환 스킵 플래그 정보에 기초하여 결정하고, 결정된 컨텍스트 모델을 기반으로 변환 스킵 플래그를 코딩하는 방안을 제안한다.

[334] 현재 블록의 상측 주변 블록의 변환 스킵 플래그 및 좌측 주변 블록의 변환 스킵 플래그에 기초하여 현재 블록의 변환 스킵 플래그에 대한 컨텍스트 모델을 도출하는 경우, 예를 들어 상기 ctxInc는 하기 표 34와 같이 0, 1 및 2 중 어느 하나로 도출될 수 있다.

[335] [표34]

Syntax element	binIdx					
	0	1	2	3	4	>= 5
transform_skip_flag[ ][ ]	0,1,2	na	na	na	na	na

[336] 이 때, 상기 ctxInc를 도출하는 일 실시예는 상술한 표 28 및 하기 표 35와 같이 정의될 수 있다. 표 28 및 표 35에 따르면, 현재 블록의 상측 주변 블록의 변환 스킵 플래그 정보 및 좌측 주변 블록 각각의 변환 스킵 플래그에 기초하여 상기 ctxInc가 도출될 수 있다.

[337] [표35]

Syntax element	condL	condA	ctxSetIdx
alf_ctb_flag[x0][y0][cIdx]	alf_ctb_flag[xNbL][yNbL][cIdx]	alf_ctb_flag[xNbA][yNbA][cIdx]	cIdx
qsplit_qt_flag	cqtDepth[xNbL][yNbL] > cqtDepth	cqtDepth[xNbA][yNbA] > cqtDepth	(cqtDepth < 2) ? 0 : 1
msplit_cu_flag	CbHeight[xNbL][yNbL] < cbHeight	CbWidth[xNbA][yNbA] < cbWidth	(allowSplitBtVer + allowSplitBtHor + allowSplitTtVer + allowSplitTtHor + 2 * allowSplitQt − 1) / 3
cu_skip_flag[x0][y0]	cu_skip_flag[xNbL][yNbL]	cu_skip_flag[xNbA][yNbA]	0
pred_mode_flag[x0][y0]	CuPredMode[xNbL][yNbL] = = MODE_INTRA	CuPredMode[xNbA][yNbA] = = MODE_INTRA	0
pred_mode_ibc_flag[x0][y0]	CuPredMode[xNbL][yNbL] = = MODE_IBC	CuPredMode[xNbA][yNbA] = = MODE_IBC	0
amvr_flag[x0][y0]	amvr_flag[xNbL][yNbL]	amvr_flag[xNbA][yNbA]	0
merge_subblock_flag[x0][y0]	merge_subblock_flag[xNbL][yNbL] | inter_affine_flag[xNbL][yNbL]	merge_subblock_flag[xNbA][yNbA] | inter_affine_flag[xNbA][yNbA]	0
merge_triangle_flag[x0][y0]	merge_triangle_flag [xNbL][yNbL]	merge_triangle_flag [xNbA][yNbA]	0
inter_affine_flag [x0][y0]	merge_subblock_flag[xNbL][yNbL] | inter_affine_flag[xNbL][yNbL]	merge_subblock_flag[xNbA][yNbA] | inter_affine_flag[xNbA][yNbA]	0
tramsform_skip_flag [x0][y0]	transform_skip_flag [xNbL][yNbL]	transform_skip_flag [xNbA][yNbA]	0

[338] 상술한 표 28의 수학식 (28-2)및 표 35를 참조하면, 상기 ctxInc는 현재 블록의 상측 주변 블록의 변환 스킵 플래그 정보 및 좌측 주변 블록의 변환 스킵 플래그가 모두 1이 아닌 경우에는 '0'으로, 상측 주변 블록의 변환 스킵 플래그 정보 및 좌측 주변 블록의 변환 스킵 플래그 중 어느 하나가 1인 경우에는 '1'로 도출될 수 있고, 상측 주변 블록의 변환 스킵 플래그 정보 및 좌측 주변 블록의 변환 스킵 플래그 모두 1인 경우에는 '2'로 도출될 수 있다.

[339] 따라서, 상기 상측 주변 블록의 변환 스킵 플래그 및 상기 좌측 주변 블록의 변환 스킵 플래그가 모두 1이 아닌 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 0으로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 1 ctxIdx로 도출(즉, 제 1 컨텍스트 모델로 도출)될 수 있다. 또한, 상기 상측 주변 블록의 변환 스킵 플래그 및 상기 좌측 주변 블록의 변환 스킵 플래그 중 어느 하나가 1인 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 1로 도출될 수 있고, 상기 변환

스킵 플래그에 대한 ctxIdx 는 제 2 ctxIdx로 도출(즉, 제 2 컨텍스트 모델로 도출)될 수 있다. 또한, 상기 상측 주변 블록의 변환 스킵 플래그 및 상기 좌측 주변 블록의 변환 스킵 플래그 모두 1인 경우, 상기 변환 스킵 플래그에 대한 ctxIdc 는 2로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 3 ctxIdx로 도출(즉, 제 3 컨텍스트 모델로 도출)될 수 있다.

[340] 현재 블록의 상측 주변 블록의 변환 스킵 플래그 및 좌측 주변 블록의 변환 스킵 플래그에 기초하여 현재 블록의 변환 스킵 플래그에 대한 컨텍스트 모델을 도출하는 경우, 상기 ctxInc는 상술한 표 34와 달리 0 및 1 중 어느 하나로 도출될 수도 있으며, 이는 하기 표 36과 같다.

[341] [표36]

Syntax element	binIdx					
	0	1	2	3	4	>= 5
transform_skip_flag[ ][ ]	0,1	na	na	na	na	na

[342] 이 때, 상기 ctxInc를 도출하는 일 실시예는 상술한 표 33 및 표 35와 같이 정의될 수 있다. 표 33 및 표 35에 따르면, 현재 블록의 상측 주변 블록 의 변환 스킵 플래그 및 좌측 주변 블록 각각의 변환 스킵 플래그에 기초하여 상기 ctxInc가 도출될 수 있다.

[343] 상술한 표 33의 수학식 (33-3) 및 표 35를 참조하면, 상기 ctxInc는 현재 블록의 상측 주변 블록의 변환 스킵 플래그 정보 및 좌측 주변 블록의 변환 스킵 플래그가 모두 1이 아닌 경우에는 '0'으로, 상측 주변 블록의 변환 스킵 플래그 정보 및 좌측 주변 블록의 변환 스킵 플래그 중 적어도 하나가 1인 경우에는 '1'로 도출될 수 있다.

[344] 따라서, 상기 상측 주변 블록의 변환 스킵 플래그 및 상기 좌측 주변 블록의 변환 스킵 플래그가 모두 1이 아닌 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 0으로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 1 ctxIdx로 도출(즉, 제 1 컨텍스트 모델로 도출)될 수 있다. 또한, 상기 상측 주변 블록의 변환 스킵 플래그 및 상기 좌측 주변 블록의 변환 스킵 플래그 중 적어도 하나가 1인 경우 상기 변환 스킵 플래그에 대한 ctxIdc 는 1로 도출될 수 있고, 상기 변환 스킵 플래그에 대한 ctxIdx 는 제 2 ctxIdx로 도출(즉, 제 2 컨텍스트 모델로 도출)될 수 있다.

[345] 도 6은 본 문서에 따른 인코딩 장치에 의한 영상 인코딩 방법을 개략적으로 나타낸다. 도 6에서 개시된 방법은 도 2에서 개시된 인코딩 장치에 의하여 수행될 수 있다. 예를 들어, 도 6의 S610 내지 S630은 상기 인코딩 장치의 엔트로피 인코딩부에 의하여 수행될 수 있다. 또한, 비록 도시되지는 않았으나 예측 샘플을 도출하는 과정은 상기 인코딩 장치의 예측부에 의하여 수행될 수 있고, 상기 현재 블록에 대한 원본 샘플과 예측 샘플을 기반으로 상기 현재 블록에 대한 레지듀얼 샘플을 도출하는 과정은 상기 인코딩 장치의 감산부에 의하여 수행될 수 있고, 상기 현재 블록에 대한 레지듀얼 샘플과 예측 샘플을

기반으로 상기 현재 블록에 대한 복원 샘플 및 복원 픽처를 생성하는 과정은 상기 인코딩 장치의 가산부에 의하여 수행될 수 있다.

[346] 인코딩 장치는 현재 블록에 변환 스킵이 적용되었는지 여부를 나타내는 변환 스킵 플래그(transform skip flag)에 대한 컨텍스트 모델을 도출할 수 있다(S610).

[347] 먼저, 인코딩 장치는 현재 블록에 인터 예측을 수행할지 또는 인트라 예측을 수행할지 여부를 결정할 수 있고, 구체적인 인터 예측 모드 또는 구체적인 인트라 예측 모드를 RD 코스트 기반으로 결정할 수 있다. 결정된 모드에 따라 인코딩 장치는 상기 현재 블록에 대한 예측 샘플을 도출할 수 있고, 상기 현재 블록에 대한 원본 샘플과 상기 예측 샘플의 감산을 통하여 상기 레지듀얼 샘플을 도출할 수 있다.

[348] 이후, 인코딩 장치는 상기 현재 블록에 대하여 변환이 적용되는지 여부를 결정할 수 있다. 즉, 인코딩 장치는 상기 현재 블록의 상기 레지듀얼 샘플에 대하여 변환이 적용되는지 여부를 결정할 수 있다. 인코딩 장치는 코딩 효율을 고려하여 상기 현재 블록에 대한 변환 적용 여부를 결정할 수 있다. 예를 들어, 인코딩 장치는 상기 현재 블록에 대하여 변환이 적용되지 않는 것으로 결정할 수 있다.

[349] 인코딩 장치는 레지듀얼 샘플 및 상기 변환 스킵이 적용되는지 여부를 기반으로 레지듀얼 정보를 생성할 수 있다. 레지듀얼 정보는 상기 현재 블록에 대한 변환 스킵 플래그를 포함할 수 있다. 상기 변환 스킵 플래그는 상기 현재 블록에 변환 스킵이 적용되는지 여부를 나타낼 수 있다. 상기 변환 스킵 플래그를 나타내는 신텍스 엘리먼트는 상술한 transform_skip_flag 일 수 있다.

[350] 예를 들어, 상기 변환 스킵 플래그에 대한 상기 컨텍스트 모델은 상기 변환 스킵 플래그에 대한 컨텍스트 인덱스 증분(context index increment)을 기반으로 결정될 수 있다. 예를 들어, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은, 상기 현재 블록의 예측 모드 정보 또는 상기 현재 블록의 주변 블록들 중 적어도 하나에 기초하여 도출될 수 있다.

[351] 일 예로, 상기 현재 블록의 예측 모드가 IBC(Intra block copy) 예측 모드가 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고, 상기 현재 블록의 예측 모드가 IBC 예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 1로 도출될 수 있다.

[352] 일 예로, 상기 주변 블록들은 상기 현재 블록의 상측 주변 블록 및 좌측 주변 블록을 포함하고, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 상기 상측 주변 블록의 예측 모드 정보 및 상기 좌측 주변 블록의 예측 모드 정보를 기반으로 도출되고, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드가 모두 IBC(Intra block copy) 예측 모드가 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드 중 어느 하나가 IBC예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 1로 도출되고,

[353] 상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드가 모두 IBC예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 2로 도출될 수 있다.

[354] 일 예로, 상기 주변 블록들은 상기 현재 블록의 상측 주변 블록 및 좌측 주변 블록을 포함하고, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 상기 상측 주변 블록의 예측 모드 정보 및 상기 좌측 주변 블록의 예측 모드 정보를 기반으로 도출되고, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드가 모두 IBC(Intra block copy) 예측 모드가 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드 중 적어도 하나가 IBC예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 1로 도출될 수 있다.

[355] 일 예로, 상기 주변 블록들은 상기 현재 블록의 상측 주변 블록 및 좌측 주변 블록을 포함하고, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은, 상기 상측 주변 블록의 변환 스킵 플래그 정보 및 상기 좌측 주변 블록의 변환 스킵 플래그 정보에 기초하여 도출될 수 있다.

[356] 일 예로, 상기 상측 주변 블록의 변환 스킵 플래그 및 상기 좌측 주변 블록의 변환 스킵 플래그가 모두 1이 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고, 상기 상측 주변 블록의 상기 변환 스킵 플래그 및 상기 좌측 주변 블록의 상기 변환 스킵 플래그 중 어느 하나가 1인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 컨택스트 모델 1로 도출되고, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 변환 스킵 플래그 모두 1인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 2로 도출될 수 있다.

[357] 일 예로, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 변환 스킵 플래그가 모두 1이 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 변환 스킵 플래그 중 적어도 하나가 IBC예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 1로 도출될 수 있다.

[358] 일 예로, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 변환 스킵 플래그가 모두 1이 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 모델은 컨택스트 모델 0으로 도출되고, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 변환 스킵 플래그 중 적어도 하나가 1인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 모델은 컨택스트 모델 1로 도출될 수 있다.

[359] 인코딩 장치는 상기 컨택스트 모델을 기반으로 상기 변환 스킵 플래그를 인코딩할 수 있다(S620).

[360] 인코딩 장치는 상기 인코딩된 변환 스킵 플래그를 포함하는 인코딩된 영상 정보를 출력할 수 있다(S630).

[361] 예를 들어, 인코딩 장치는 상기 변환 스킵 플래그를 포함하는 레지듀얼 정보를 포함하는 영상 정보를 비트스트림으로 출력할 수 있다. 상기 비트스트림은 레지듀얼 정보를 포함할 수 있다.

[362] 한편, 상기 비트스트림은 네트워크 또는 (디지털) 저장매체를 통하여 디코딩 장치로 전송될 수 있다. 여기서 네트워크는 방송망 및/또는 통신망 등을 포함할 수 있고, 디지털 저장매체는 USB, SD, CD, DVD, 블루레이, HDD, SSD 등 다양한 저장매체를 포함할 수 있다. 또는, 상기 비트스트림은 컴퓨터 판독 가능한 저장매체에 저장될 수 있다. 예를 들어, 비트스트림은 영상 정보 또는 비디오 정보로 나타낼 수도 있다.

[363] 도 7은 본 문서에 따른 영상 인코딩 방법을 수행하는 인코딩 장치를 개략적으로 나타낸다. 도 6에서 개시된 방법은 도 7에서 개시된 인코딩 장치에 의하여 수행될 수 있다. 구체적으로 예를 들어, 도 7의 상기 인코딩 장치의 엔트로피 인코딩부는 도 6의 S610 내지 S630을 수행할 수 있다. 또한, 비록 도시되지는 않았으나 예측 샘플을 도출하는 과정은 상기 인코딩 장치의 예측부에 의하여 수행될 수 있고, 상기 현재 블록에 대한 레지듀얼 샘플과 예측 샘플을 기반으로 상기 현재 블록에 대한 복원 샘플을 도출하는 과정은 상기 인코딩 장치의 가산부에 의하여 수행될 수 있고, 상기 현재 블록에 대한 예측 정보를 인코딩하는 과정은 상기 인코딩 장치의 엔트로피 인코딩부에 의하여 수행될 수 있다.

[364] 도 8은 본 문서에 따른 디코딩 장치에 의한 영상 디코딩 방법을 개략적으로 나타낸다. 도 8에서 개시된 방법은 도 3에서 개시된 디코딩 장치에 의하여 수행될 수 있다. 구체적으로 예를 들어, 도 8의 S810 내지 S830은 상기 디코딩 장치의 엔트로피 디코딩부에 의하여 수행될 수 있고, S840은 상기 디코딩 장치의 레지듀얼 처리부에 의하여 수행될 수 있고, S850은 상기 디코딩 장치의 가산부에 의하여 수행될 수 있다. 또한, 비록 도시되지는 않았으나 예측 샘플을 도출하는 과정은 상기 디코딩 장치의 예측부에 의하여 수행될 수 있다.

[365] 디코딩 장치는 변환 스킵 플래그를 포함하는 영상 정보를 수신 한다(S810). 디코딩 장치는 비트스트림을 통하여 상기 현재 블록에 대한 레지듀얼 정보를 포함하는 영상 정보를 수신할 수 있다. 여기서, 상기 현재 블록은 코딩 블록(Coding Block, CB) 또는 변환 블록(Transform Block, TB)일 수 있다.

[366] 예를 들어, 상기 영상 정보는 상기 현재 블록에 대한 변환 스킵 플래그를 포함할 수 있다. 예를 들어, 상기 레지듀얼 정보는 상기 현재 블록에 대한 변환 스킵 플래그를 포함할 수 있다. 상기 변환 스킵 플래그는 상기 현재 블록에 변환 스킵이 적용되는지 여부를 나타낼 수 있다. 또는 상기 변환 스킵 플래그는 transform_skip_flag 신택스 요소로 나타낼 수도 있다. 예를 들어, 상기 transform_skip_flag 신택스 요소의 값이 1인 경우, 상기 현재 블록에 변환 스킵이 적용될 수 있고, 0인 경우, 상기 현재 블록에 변환 스킵이 적용되지 않을 수 있다. 또는 설정에 따라 상기 transform_skip_flag 신택스 요소의 값이 0인 경우, 상기

현재 블록에 변환 스킵이 적용될 수 있고, 1인 경우, 상기 현재 블록에 변환 스킵이 적용되지 않을 수 있다.

[367] 디코딩 장치는 현재 블록에 변환 스킵이 적용되었는지 여부를 나타내는 상기 변환 스킵 플래그(transform skip flag)에 대한 컨텍스트 모델을 도출할 수 있다(S820).

[368] 예를 들어, 상기 변환 스킵 플래그에 대한 상기 컨텍스트 모델은 상기 변환 스킵 플래그에 대한 컨텍스트 인덱스 증분(context index increment)을 기반으로 결정될 수 있다. 예를 들어, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은, 상기 현재 블록의 예측 모드 정보 또는 상기 현재 블록의 주변 블록들 중 적어도 하나에 기초하여 도출될 수 있다.

[369] 일 예로, 상기 현재 블록의 예측 모드가 IBC(Intra block copy) 예측 모드가 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고, 상기 현재 블록의 예측 모드가 IBC 예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 1로 도출될 수 있다.

[370] 일 예로, 상기 주변 블록들은 상기 현재 블록의 상측 주변 블록 및 좌측 주변 블록을 포함하고, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 상기 상측 주변 블록의 예측 모드 정보 및 상기 좌측 주변 블록의 예측 모드 정보를 기반으로 도출되고, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드가 모두 IBC(Intra block copy) 예측 모드가 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드 중 어느 하나가 IBC예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 1로 도출되고,

[371] 상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드가 모두 IBC예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 2로 도출될 수 있다.

[372] 일 예로, 상기 주변 블록들은 상기 현재 블록의 상측 주변 블록 및 좌측 주변 블록을 포함하고, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 상기 상측 주변 블록의 예측 모드 정보 및 상기 좌측 주변 블록의 예측 모드 정보를 기반으로 도출되고, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드가 모두 IBC(Intra block copy) 예측 모드가 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드 중 적어도 하나가 IBC예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 1로 도출될 수 있다.

[373] 일 예로, 상기 주변 블록들은 상기 현재 블록의 상측 주변 블록 및 좌측 주변 블록을 포함하고, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은, 상기 상측 주변 블록의 변환 스킵 플래그 정보 및 상기 좌측 주변 블록의 변환 스킵 플래그 정보에 기초하여 도출될 수 있다.

[374] 일 예로, 상기 상측 주변 블록의 변환 스킵 플래그 및 상기 좌측 주변 블록의 변환 스킵 플래그가 모두 1이 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고, 상기 상측 주변 블록의 상기 변환 스킵 플래그 및 상기 좌측 주변 블록의 상기 변환 스킵 플래그 중 어느 하나가 1인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 컨텍스트 모델 1로 도출되고, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 변환 스킵 플래그 모두 1인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 2로 도출될 수 있다.

[375] 일 예로, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 변환 스킵 플래그가 모두 1이 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 변환 스킵 플래그 중 적어도 하나가 IBC예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 1로 도출될 수 있다.

[376] 일 예로, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 변환 스킵 플래그가 모두 1이 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨텍스트 모델은 컨텍스트 모델 0으로 도출되고, 상기 상측 주변 블록 및 상기 좌측 주변 블록의 변환 스킵 플래그 중 적어도 하나가 1인 경우, 상기 변환 스킵 플래그에 대한 상기 컨텍스트 모델은 컨텍스트 모델 1로 도출될 수 있다.

[377] 디코딩 장치는 상기 컨텍스트 모델을 기반으로 상기 변환 스킵 플래그를 디코딩할 수 있다(S830).

[378] 디코딩 장치는 상기 디코딩된 변환 스킵 플래그를 기반으로 레지듀얼 샘플을 도출할 수 있다(S840).

[379] 예를 들어, 상기 transform_skip_flag 신택스 요소의 값이 1인 경우, 상기 현재 블록에 대한 레지듀얼 신호(또는 레지듀얼에 관한 정보)는 변환 없이 픽셀 도메인(공간 도메인) 상에서 시그널링될 수 있다. 또는 상기 transform_skip_flag 신택스 요소의 값이 0인 경우, 상기 현재 블록에 대한 레지듀얼 신호(또는 레지듀얼에 관한 정보)는 변환이 수행되어 변환 도메인 상에서 시그널링될 수 있다. 예를 들어, 디코딩 장치는 상기 변환 없이 또는 변환이 수행되어 시그널링된 레지듀얼 신호를 기반으로 레지듀얼 샘플들을 도출할 수 있다.

[380] 디코딩 장치는 상기 레지듀얼 샘플을 기반으로 복원 픽처를 생성할 수 있다(S850).

[381] 예를 들어, 디코딩 장치는 비트스트림을 통하여 수신된 예측 정보를 기반으로 상기 현재 블록에 대한 인터 예측 모드 또는 인트라 예측 모드를 수행하여 예측 샘플을 도출할 수 있고, 상기 예측 샘플과 상기 레지듀얼 샘플의 가산을 통하여 상기 복원 픽처를 생성할 수 있다. 또한, 예를 들어, 상기 예측 정보는 상기 현재 블록의 인트라 예측 모드를 나타내는 정보를 포함할 수 있다. 디코딩 장치는 상기 현재 블록의 인트라 예측 모드를 나타내는 정보를 기반으로 상기 현재 블록의 상기 인트라 예측 모드를 도출할 수 있고, 상기 현재 블록의 참조 샘플들

및 상기 인트라 예측 모드를 기반으로 상기 현재 블록의 예측 샘플을 도출할 수 있다. 상기 참조 샘플들은 상기 현재 블록의 상측 참조 샘플들 및 좌측 참조 샘플들을 포함할 수 있다. 예를 들어, 상기 현재 블록의 사이즈가 NxN이고, 상기 현재 블록의 좌상단(top-left) 샘플 포지션의 x성분이 0 및 y성분이 0인 경우, 상기 좌측 참조 샘플들은 p[-1][0] 내지 p[-1][2N-1], 상기 상측 참조 샘플들은 p[0][-1] 내지 p[2N-1][-1]일 수 있다.

[382] 이후 필요에 따라 주관적/객관적 화질을 향상시키기 위하여 디블록킹 필터링, SAO 및/또는 ALF 절차와 같은 인루프 필터링 절차가 상기 복원 픽처에 적용될 수 있음은 상술한 바와 같다.

[383] 도 9는 본 문서에 따른 영상 디코딩 방법을 수행하는 디코딩 장치를 개략적으로 나타낸다. 도 8에서 개시된 방법은 도 9에서 개시된 디코딩 장치에 의하여 수행될 수 있다. 구체적으로 예를 들어, 도 9의 상기 디코딩 장치의 엔트로피 디코딩부는 도 8의 S810 내지 S830을 수행할 수 있고, 도 9의 상기 디코딩 장치의 레지듀얼 처리부는 도 8의 S840을 수행할 수 있고, 도 9의 상기 디코딩 장치의 가산부는 도 8의 S850을 수행할 수 있다. 또한, 비록 도시되지는 않았으나 예측 샘플을 도출하는 과정은 도 9의 상기 디코딩 장치의 예측부에 의하여 수행될 수 있다.

[384] 상술한 실시예에서, 방법들은 일련의 단계 또는 블록으로써 순서도를 기초로 설명되고 있지만, 본 문서는 단계들의 순서에 한정되는 것은 아니며, 어떤 단계는 상술한 바와 다른 단계와 다른 순서로 또는 동시에 발생할 수 있다. 또한, 당업자라면 순서도에 나타내어진 단계들이 배타적이지 않고, 다른 단계가 포함되거나 순서도의 하나 또는 그 이상의 단계가 본 문서의 범위에 영향을 미치지 않고 삭제될 수 있음을 이해할 수 있을 것이다.

[385] 본 문서에서 설명한 실시예들은 프로세서, 마이크로 프로세서, 컨트롤러 또는 칩 상에서 구현되어 수행될 수 있다. 예를 들어, 각 도면에서 도시한 기능 유닛들은 컴퓨터, 프로세서, 마이크로 프로세서, 컨트롤러 또는 칩 상에서 구현되어 수행될 수 있다. 이 경우 구현을 위한 정보(ex. information on instructions) 또는 알고리즘이 디지털 저장 매체에 저장될 수 있다.

[386] 또한, 본 문서의 실시예들이 적용되는 디코딩 장치 및 인코딩 장치는 멀티미디어 방송 송수신 장치, 모바일 통신 단말, 홈 시네마 비디오 장치, 디지털 시네마 비디오 장치, 감시용 카메라, 비디오 대화 장치, 비디오 통신과 같은 실시간 통신 장치, 모바일 스트리밍 장치, 저장 매체, 캠코더, 주문형 비디오(VoD) 서비스 제공 장치, OTT 비디오(Over the top video) 장치, 인터넷 스트리밍 서비스 제공 장치, 3차원(3D) 비디오 장치, 화상 전화 비디오 장치, 운송 수단 단말 (ex. 차량 단말, 비행기 단말, 선박 단말 등) 및 의료용 비디오 장치 등에 포함될 수 있으며, 비디오 신호 또는 데이터 신호를 처리하기 위해 사용될 수 있다. 예를 들어, OTT 비디오(Over the top video) 장치로는 게임 콘솔, 블루레이 플레이어, 인터넷 접속 TV, 홈시어터 시스템, 스마트폰, 태블릿 PC, DVR(Digital

Video Recoder) 등을 포함할 수 있다.

[387] 또한, 본 문서의 실시예들이 적용되는 처리 방법은 컴퓨터로 실행되는 프로그램의 형태로 생산될 수 있으며, 컴퓨터가 판독할 수 있는 기록 매체에 저장될 수 있다. 본 문서에 따른 데이터 구조를 가지는 멀티미디어 데이터도 또한 컴퓨터가 판독할 수 있는 기록 매체에 저장될 수 있다. 상기 컴퓨터가 판독할 수 있는 기록 매체는 컴퓨터로 읽을 수 있는 데이터가 저장되는 모든 종류의 저장 장치 및 분산 저장 장치를 포함한다. 상기 컴퓨터가 판독할 수 있는 기록 매체는, 예를 들어, 블루레이 디스크(BD), 범용 직렬 버스(USB), ROM, PROM, EPROM, EEPROM, RAM, CD-ROM, 자기 테이프, 플로피 디스크 및 광학적 데이터 저장 장치를 포함할 수 있다. 또한, 상기 컴퓨터가 판독할 수 있는 기록 매체는 반송파(예를 들어, 인터넷을 통한 전송)의 형태로 구현된 미디어를 포함한다. 또한, 인코딩 방법으로 생성된 비트스트림이 컴퓨터가 판독할 수 있는 기록 매체에 저장되거나 유무선 통신 네트워크를 통해 전송될 수 있다.

[388] 또한, 본 문서의 실시예는 프로그램 코드에 의한 컴퓨터 프로그램 제품으로 구현될 수 있고, 상기 프로그램 코드는 본 문서의 실시예에 의해 컴퓨터에서 수행될 수 있다. 상기 프로그램 코드는 컴퓨터에 의해 판독가능한 캐리어 상에 저장될 수 있다.

[389] 도 10은 본 문서의 실시예들이 적용되는 컨텐츠 스트리밍 시스템 구조도를 예시적으로 나타낸다.

[390] 본 문서의 실시예들이 적용되는 컨텐츠 스트리밍 시스템은 크게 인코딩 서버, 스트리밍 서버, 웹 서버, 미디어 저장소, 사용자 장치 및 멀티미디어 입력 장치를 포함할 수 있다.

[391] 상기 인코딩 서버는 스마트폰, 카메라, 캠코더 등과 같은 멀티미디어 입력 장치들로부터 입력된 컨텐츠를 디지털 데이터로 압축하여 비트스트림을 생성하고 이를 상기 스트리밍 서버로 전송하는 역할을 한다. 다른 예로, 스마트폰, 카메라, 캠코더 등과 같은 멀티미디어 입력 장치들이 비트스트림을 직접 생성하는 경우, 상기 인코딩 서버는 생략될 수 있다.

[392] 상기 비트스트림은 본 문서의 실시예들이 적용되는 인코딩 방법 또는 비트스트림 생성 방법에 의해 생성될 수 있고, 상기 스트리밍 서버는 상기 비트스트림을 전송 또는 수신하는 과정에서 일시적으로 상기 비트스트림을 저장할 수 있다.

[393] 상기 스트리밍 서버는 웹 서버를 통한 사용자 요청에 기초하여 멀티미디어 데이터를 사용자 장치에 전송하고, 상기 웹 서버는 사용자에게 어떠한 서비스가 있는지를 알려주는 매개체 역할을 한다. 사용자가 상기 웹 서버에 원하는 서비스를 요청하면, 상기 웹 서버는 이를 스트리밍 서버에 전달하고, 상기 스트리밍 서버는 사용자에게 멀티미디어 데이터를 전송한다. 이때, 상기 컨텐츠 스트리밍 시스템은 별도의 제어 서버를 포함할 수 있고, 이 경우 상기 제어 서버는 상기 컨텐츠 스트리밍 시스템 내 각 장치 간 명령/응답을 제어하는

역할을 한다.

[394] 상기 스트리밍 서버는 미디어 저장소 및/또는 인코딩 서버로부터 컨텐츠를 수신할 수 있다. 예를 들어, 상기 인코딩 서버로부터 컨텐츠를 수신하게 되는 경우, 상기 컨텐츠를 실시간으로 수신할 수 있다. 이 경우, 원활한 스트리밍 서비스를 제공하기 위하여 상기 스트리밍 서버는 상기 비트스트림을 일정 시간동안 저장할 수 있다.

[395] 상기 사용자 장치의 예로는, 휴대폰, 스마트 폰(smart phone), 노트북 컴퓨터(laptop computer), 디지털방송용 단말기, PDA(personal digital assistants), PMP(portable multimedia player), 네비게이션, 슬레이트 PC(slate PC), 태블릿 PC(tablet PC), 울트라북(ultrabook), 웨어러블 디바이스(wearable device, 예를 들어, 워치형 단말기 (smartwatch), 글래스형 단말기 (smart glass), HMD(head mounted display)), 디지털 TV, 데스크탑 컴퓨터, 디지털 사이니지 등이 있을 수 있다. 상기 컨텐츠 스트리밍 시스템 내 각 서버들은 분산 서버로 운영될 수 있으며, 이 경우 각 서버에서 수신하는 데이터는 분산 처리될 수 있다.

[396] 본 명세서에 기재된 청구항들은 다양한 방식으로 조합될 수 있다. 예를 들어, 본 명세서의 방법 청구항의 기술적 특징이 조합되어 장치로 구현될 수 있고, 본 명세서의 장치 청구항의 기술적 특징이 조합되어 방법으로 구현될 수 있다. 또한, 본 명세서의 방법 청구항의 기술적 특징과 장치 청구항의 기술적 특징이 조합되어 장치로 구현될 수 있고, 본 명세서의 방법 청구항의 기술적 특징과 장치 청구항의 기술적 특징이 조합되어 방법으로 구현될 수 있다.

청구범위

[청구항 1] 디코딩 장치에 의하여 수행되는 영상 디코딩 방법에 있어서,
변환 스킵 플래그를 포함하는 영상 정보를 수신하는 단계;
현재 블록에 변환 스킵이 적용되었는지 여부를 나타내는 상기 변환 스킵 플래그(transform skip flag)에 대한 컨텍스트 모델을 도출하는 단계;
상기 컨텍스트 모델을 기반으로 상기 변환 스킵 플래그를 디코딩하는 단계;
상기 디코딩된 변환 스킵 플래그를 기반으로 레지듀얼 샘플을 도출하는 단계; 및
상기 레지듀얼 샘플을 기반으로 복원 픽처를 생성하는 단계를 포함하고,
상기 변환 스킵 플래그에 대한 상기 컨텍스트 모델은 상기 변환 스킵 플래그에 대한 컨텍스트 인덱스 증분(context index increment)을 기반으로 결정되고,
상기 변환 스킵 플래그에 대한 상기 컨텍스트 인덱스 증분은,
상기 현재 블록의 예측 모드 정보 또는 상기 현재 블록의 주변 블록들 중 적어도 하나에 기초하여 도출되는, 영상 디코딩 방법.

[청구항 2] 제 1 항에 있어서,
상기 현재 블록의 예측 모드가 IBC(Intra block copy) 예측 모드가 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨텍스트 인덱스 증분은 0으로 도출되고,
상기 현재 블록의 예측 모드가 IBC 예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨텍스트 인덱스 증분은 1로 도출되는, 영상 디코딩 방법.

[청구항 3] 제 1 항에 있어서,
상기 주변 블록들은 상기 현재 블록의 상측 주변 블록 및 좌측 주변 블록을 포함하고,
상기 변환 스킵 플래그에 대한 상기 컨텍스트 인덱스 증분은 상기 상측 주변 블록의 예측 모드 정보 및 상기 좌측 주변 블록의 예측 모드 정보를 기반으로 도출되고,
상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드가 모두 IBC(Intra block copy) 예측 모드가 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨텍스트 인덱스 증분은 0으로 도출되고,
상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드 중 어느 하나가 IBC예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨텍스트 인덱스 증분은 1로 도출되고,
상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드가 모두 IBC예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨텍스트 인덱스 증분은

2로 도출되는, 영상 디코딩 방법.

[청구항 4] 제 1 항에 있어서,
상기 주변 블록들은 상기 현재 블록의 상측 주변 블록 및 좌측 주변 블록을 포함하고,
상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 상기 상측 주변 블록의 예측 모드 정보 및 상기 좌측 주변 블록의 예측 모드 정보를 기반으로 도출되고,
상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드가 모두 IBC(Intra block copy) 예측 모드가 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고,
상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드 중 적어도 하나가 IBC예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 1로 도출되는, 영상 디코딩 방법.

[청구항 5] 제 1 항에 있어서,
상기 주변 블록들은 상기 현재 블록의 상측 주변 블록 및 좌측 주변 블록을 포함하고,
상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은,
상기 상측 주변 블록의 변환 스킵 플래그 정보 및 상기 좌측 주변 블록의 변환 스킵 플래그 정보에 기초하여 도출되는, 영상 디코딩 방법.

[청구항 6] 제 5 항에 있어서,
상기 상측 주변 블록의 변환 스킵 플래그 및 상기 좌측 주변 블록의 변환 스킵 플래그가 모두 1이 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고,
상기 상측 주변 블록의 상기 변환 스킵 플래그 및 상기 좌측 주변 블록의 상기 변환 스킵 플래그 중 어느 하나가 1인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 1로 도출되고,
상기 상측 주변 블록 및 상기 좌측 주변 블록의 변환 스킵 플래그 모두 1인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 2로 도출되는, 영상 디코딩 방법.

[청구항 7] 제 5 항에 있어서,
상기 상측 주변 블록 및 상기 좌측 주변 블록의 변환 스킵 플래그가 모두 1이 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고,
상기 상측 주변 블록 및 상기 좌측 주변 블록의 변환 스킵 플래그 중 적어도 하나가 1인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 1로 도출되는, 영상 디코딩 방법.

[청구항 8] 인코딩 장치에 의하여 수행되는 영상 인코딩 방법에 있어서,
현재 블록에 변환 스킵이 적용되었는지 여부를 나타내는 변환 스킵

플래그(transform skip flag)에 대한 컨텍스트 모델을 도출하는 단계;
상기 컨텍스트 모델을 기반으로 상기 변환 스킵 플래그를 인코딩하는 단계; 및
상기 인코딩된 변환 스킵 플래그를 포함하는 인코딩된 영상 정보를 출력하는 단계를 포함하고,
상기 변환 스킵 플래그에 대한 상기 컨텍스트 모델은 상기 변환 스킵 플래그에 대한 컨텍스트 인덱스 증분(context index increment)을 기반으로 결정되고,
상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은,
상기 현재 블록의 예측 모드 정보 또는 상기 현재 블록의 주변 블록들 중 적어도 하나에 기초하여 도출되는, 영상 인코딩 방법.

[청구항 9] 제 8 항에 있어서,
상기 현재 블록의 예측 모드가 IBC(Intra block copy) 예측 모드가 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고,
상기 현재 블록의 예측 모드가 IBC 예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 1로 도출되는, 영상 인코딩 방법.

[청구항 10] 제 8 항에 있어서,
상기 주변 블록들은 상기 현재 블록의 상측 주변 블록 및 좌측 주변 블록을 포함하고,
상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 상기 상측 주변 블록의 예측 모드 정보 및 상기 좌측 주변 블록의 예측 모드 정보를 기반으로 도출되고,
상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드가 모두 IBC(Intra block copy) 예측 모드가 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고,
상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드 중 어느 하나가 IBC예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 1로 도출되고
상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드가 모두 IBC예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 2로 도출되는, 영상 인코딩 방법.

[청구항 11] 제 8 항에 있어서,
상기 주변 블록들은 상기 현재 블록의 상측 주변 블록 및 좌측 주변 블록을 포함하고,
상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 상기 상측 주변 블록의 예측 모드 정보 및 상기 좌측 주변 블록의 예측 모드 정보를

기반으로 도출되고,
상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드가 모두 IBC(Intra block copy) 예측 모드가 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고,
상기 상측 주변 블록 및 상기 좌측 주변 블록의 예측 모드 중 적어도 하나가 IBC예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 1로 도출되는, 영상 인코딩 방법.

[청구항 12] 제 8 항에 있어서,
상기 주변 블록들은 상기 현재 블록의 상측 주변 블록 및 좌측 주변 블록을 포함하고,
상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은,
상기 상측 주변 블록의 변환 스킵 플래그 정보 및 상기 좌측 주변 블록의 변환 스킵 플래그 정보에 기초하여 도출되는, 영상 인코딩 방법.

[청구항 13] 제 12 항에 있어서,
상기 상측 주변 블록의 변환 스킵 플래그 및 상기 좌측 주변 블록의 변환 스킵 플래그가 모두 1이 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고,
상기 상측 주변 블록의 상기 변환 스킵 플래그 및 상기 좌측 주변 블록의 상기 변환 스킵 플래그 중 어느 하나가 1인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 1로 도출되고
상기 상측 주변 블록 및 상기 좌측 주변 블록의 변환 스킵 플래그 모두 1인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 2로 도출되는, 영상 인코딩 방법.

[청구항 14] 제 12 항에 있어서,
상기 상측 주변 블록 및 상기 좌측 주변 블록의 변환 스킵 플래그가 모두 1이 아닌 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 0으로 도출되고,
상기 상측 주변 블록 및 상기 좌측 주변 블록의 변환 스킵 플래그 중 적어도 하나가 IBC예측 모드인 경우, 상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은 1로 도출되는, 영상 인코딩 방법.

[청구항 15] 디코딩 장치로 하여금 영상 디코딩 방법을 수행하도록 야기하는 영상 정보를 포함하는 비트스트림이 저장된 컴퓨터 판독가능 디지털 저장 매체에 있어서, 상기 영상 디코딩 방법은,
변환 스킵 플래그를 포함하는 영상 정보를 수신하는 단계;
현재 블록에 변환 스킵이 적용되었는지 여부를 나타내는 상기 변환 스킵 플래그(transform skip flag)에 대한 컨텍스트 모델을 도출하는 단계;
상기 컨텍스트 모델을 기반으로 상기 변환 스킵 플래그를 디코딩하는 단계;

상기 디코딩된 변환 스킵 플래그를 기반으로 레지듀얼 샘플을 도출하는 단계; 및

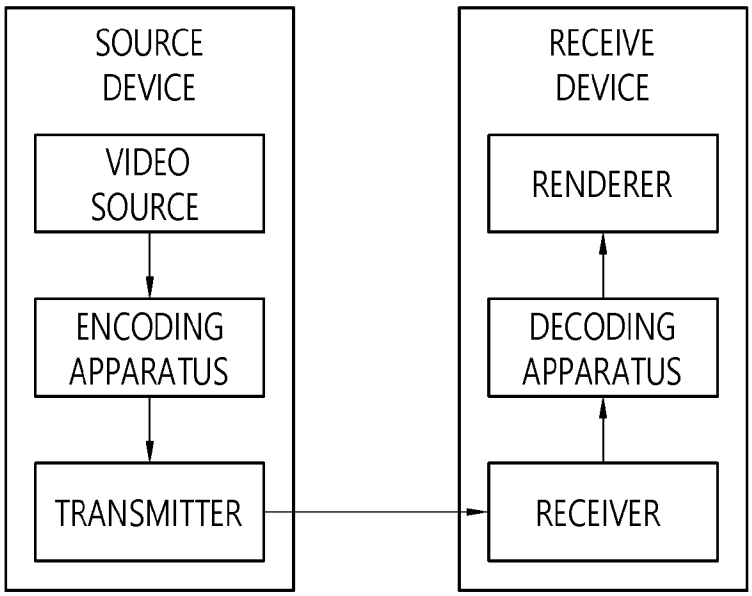
상기 레지듀얼 샘플을 기반으로 복원 픽처를 생성하는 단계를 포함하고,

상기 변환 스킵 플래그에 대한 상기 컨텍스트 모델은 상기 변환 스킵 플래그에 대한 컨텍스트 인덱스 증분(context index increment)을 기반으로 결정되고,

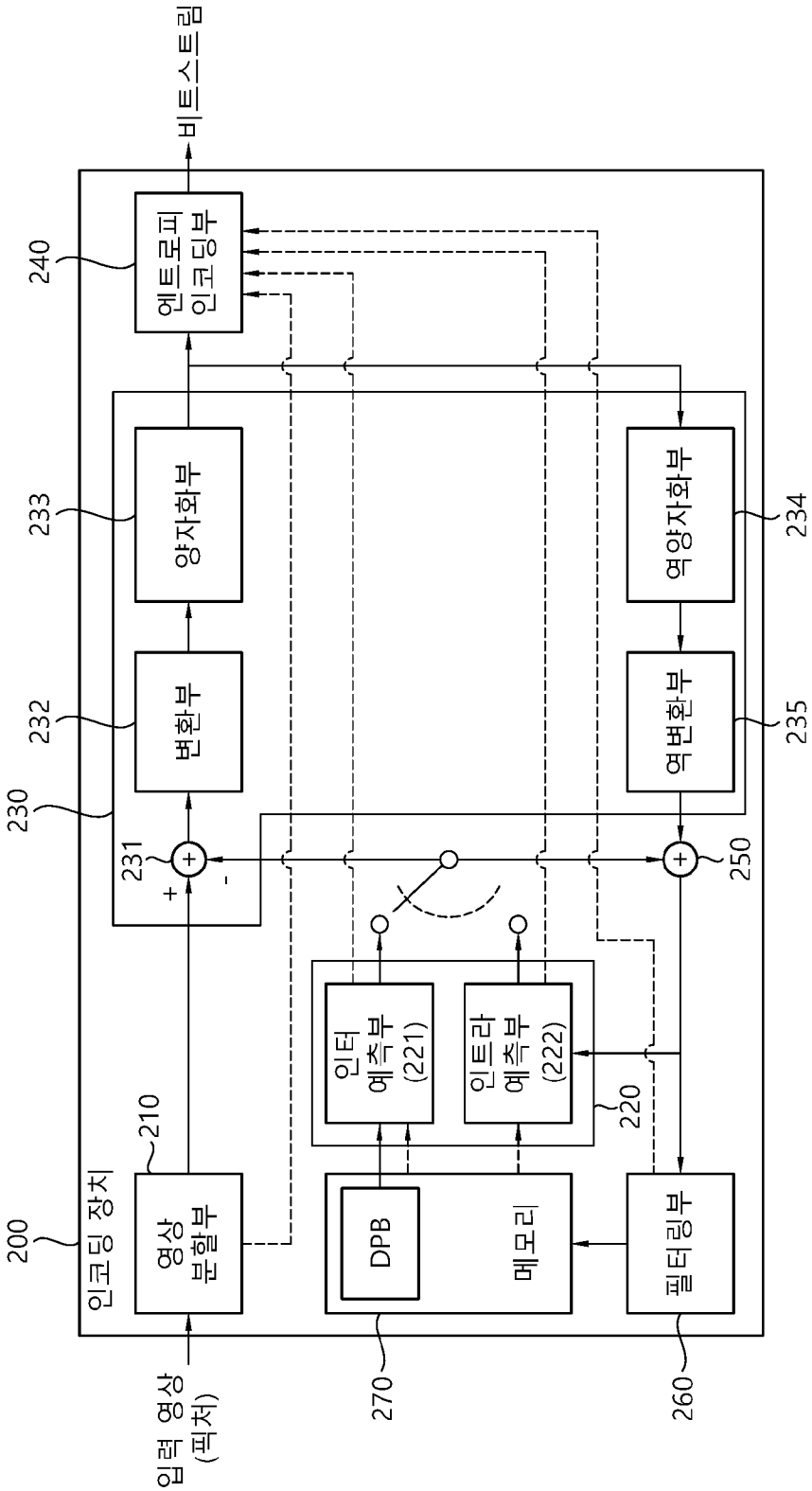
상기 변환 스킵 플래그에 대한 상기 컨택스트 인덱스 증분은,

상기 현재 블록의 예측 모드 정보 또는 상기 현재 블록의 주변 블록들 중 적어도 하나에 기초하여 도출되는, 컴퓨터 판독가능 디지털 저장 매체.

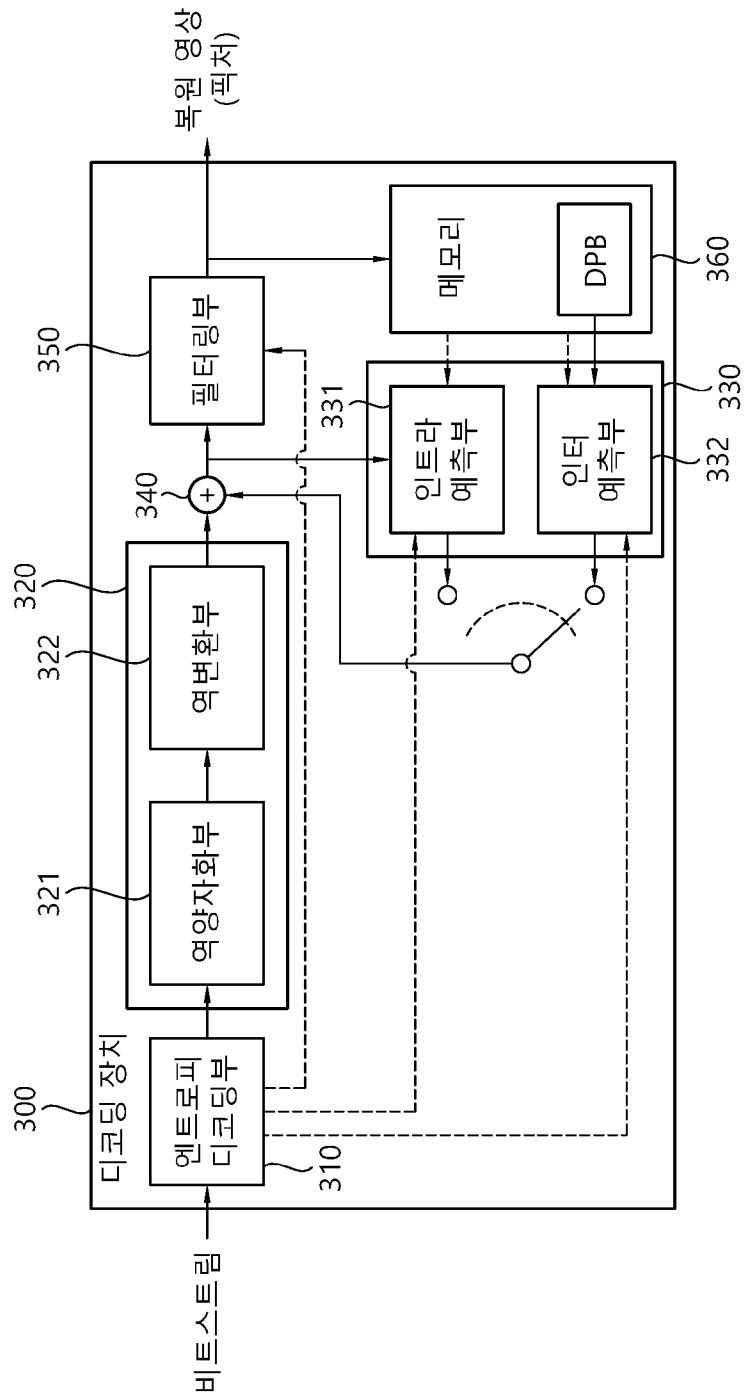
[도1]


SOURCE DEVICE
VIDEO SOURCE
ENCODING APPARATUS
TRANSMITTER
RECEIVE DEVICE
RENDERER
DECODING APPARATUS
RECEIVER

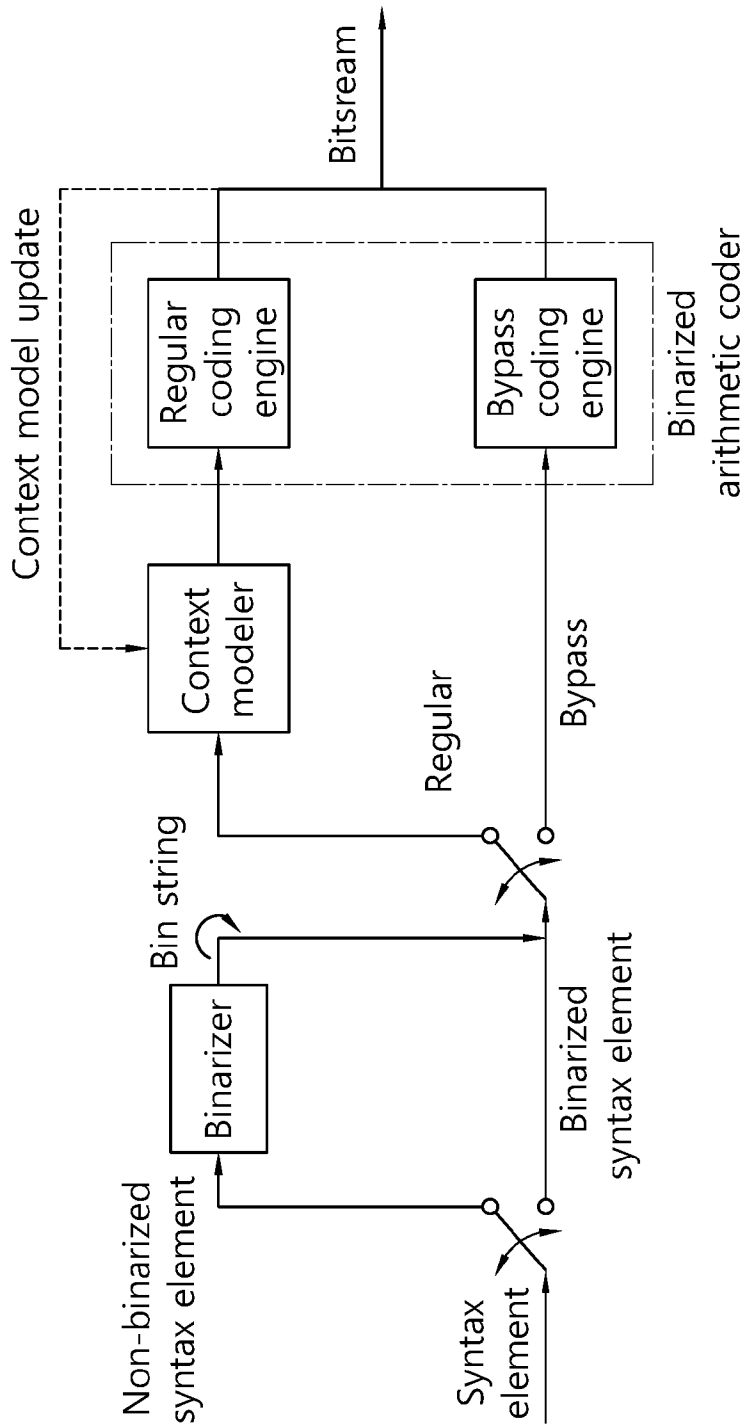
[도2]


비트스트림
엔트로피 인코딩부
240
양자화부
233
변환부
232
230
231
+
-
역양자화부
234
역변환부
235
250
인터 예측부 (221)
인트라 예측부 (222)
220
인코딩 장치
200
210
영상 분할부
DPB
메모리
필터링부
입력 영상 (픽처)
270
260

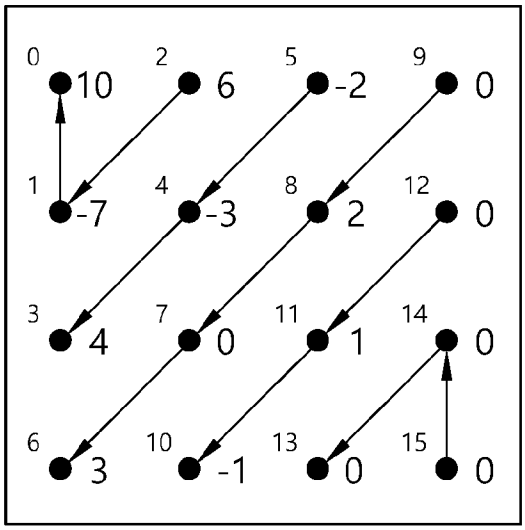
[도3]


복원 영상
(픽처)
메모리
DPB
360
350
필터링부
331
인트라
예측부
인터
예측부
330
332
340
320
322
역변환부
321
역양자화부
300
디코딩 장치
엔트로피
디코딩부
310
비트스트림

[도4]


Bitsream
Context model update
Regular coding engine
Bypass coding engine
Binarized arithmetic coder
Context modeler
Regular
Bypass
Bin string
Binarizer
Binarized syntax element
Non-binarized syntax element
Syntax element

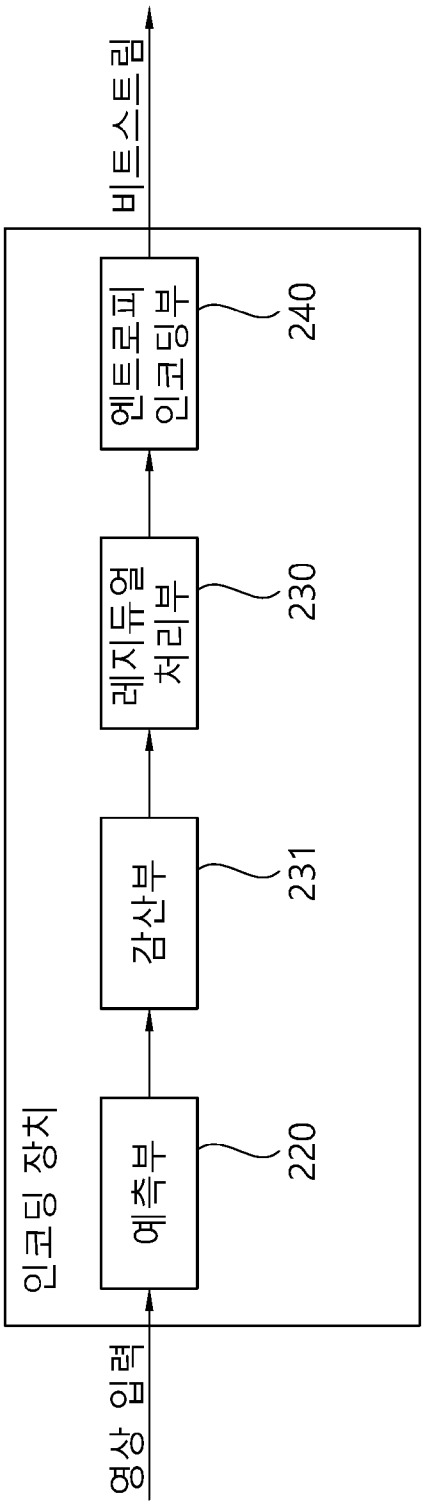
[도5]


0
10
2
6
5
-2
9
0
1
-7
4
-3
8
2
12
0
3
4
7
0
11
1
14
0
6
3
10
-1
13
0
15
0

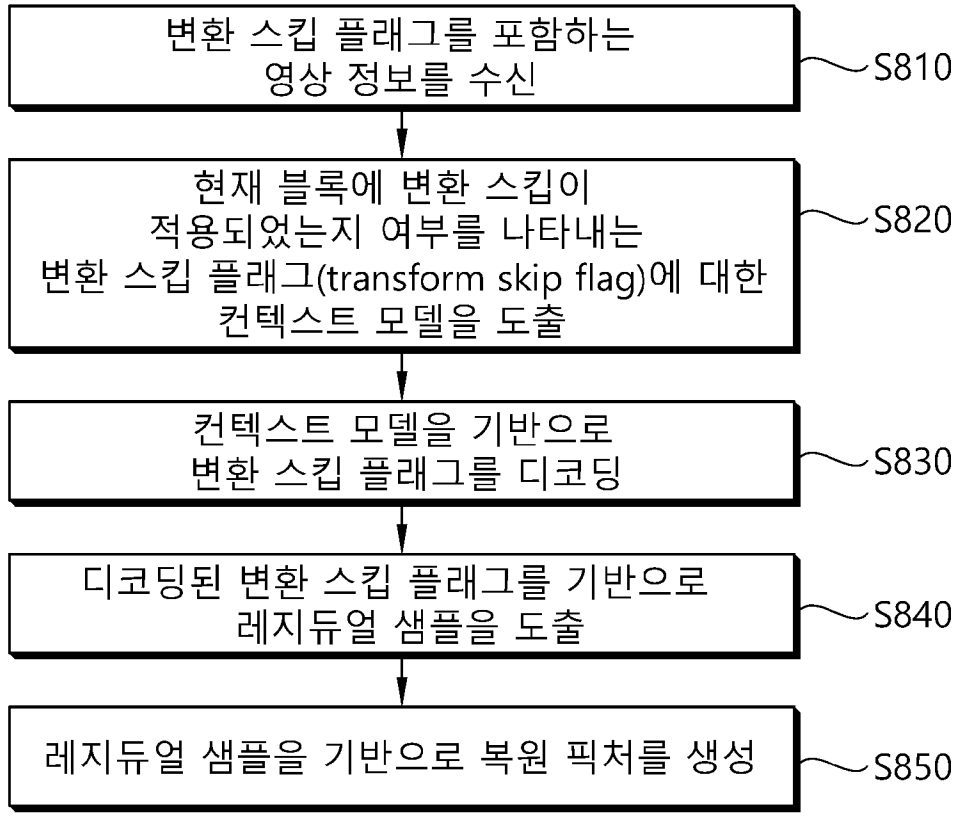
[도6]


현재 블록에 변환 스킵이
적용되었는지 여부를 나타내는
변환 스킵 플래그(transform skip flag)에 대한
컨텍스트 모델을 도출
S610
컨텍스트 모델을 기반으로
변환 스킵 플래그를 인코딩
S620
인코딩된 변환 스킵 플래그를 포함하는
인코딩된 영상 정보를 출력
S630

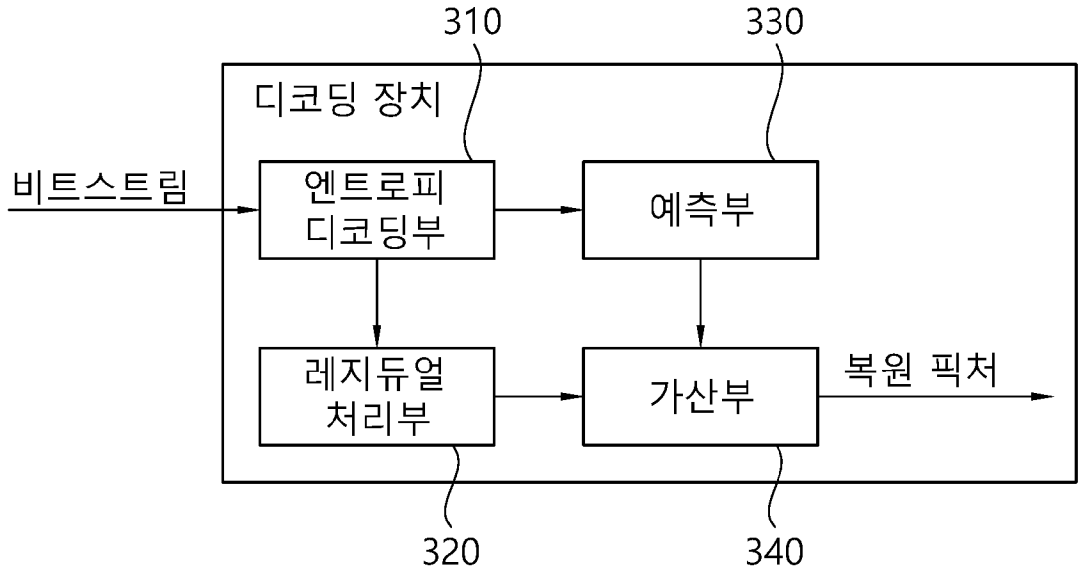
[도7]



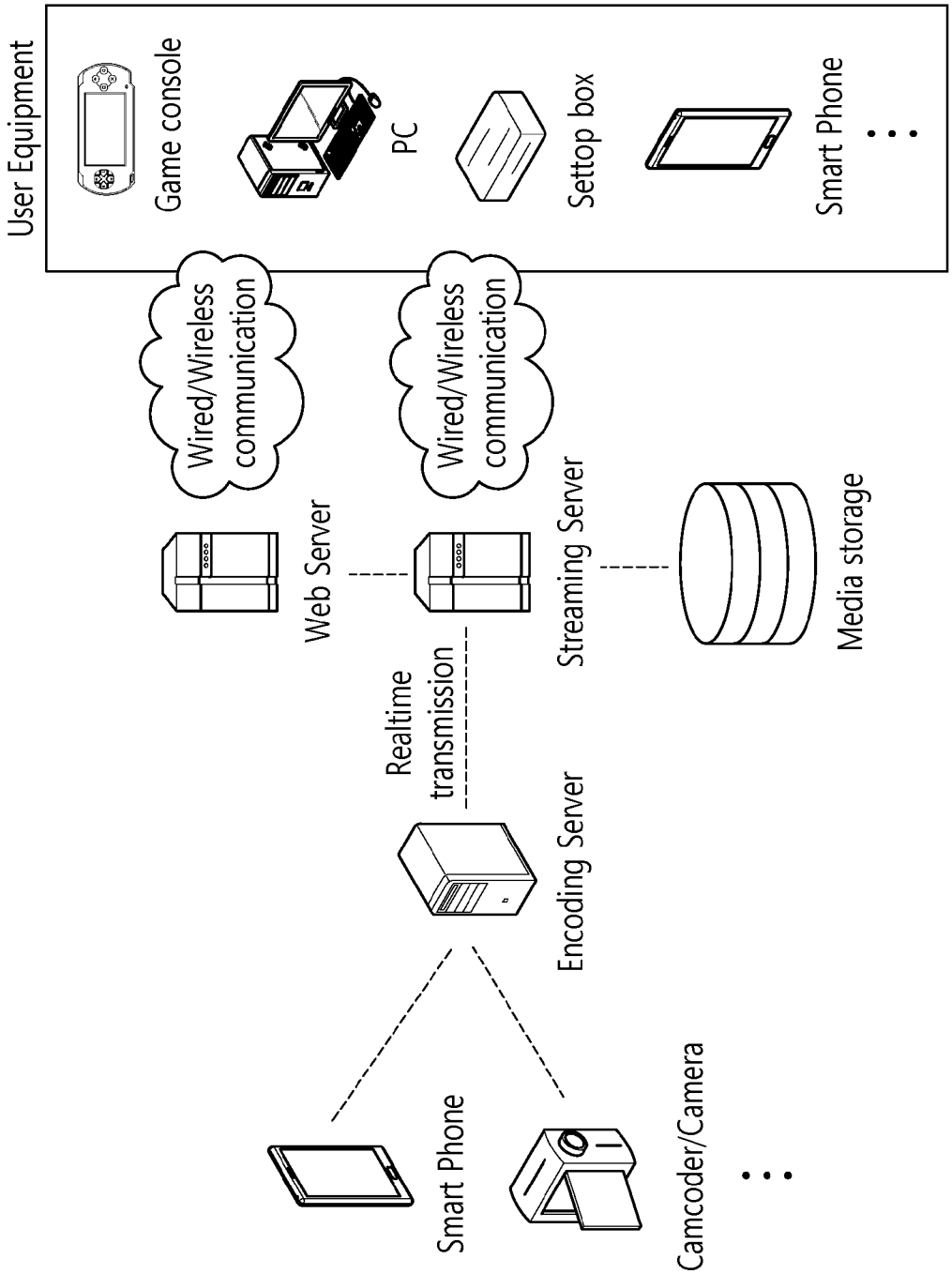
[도8]


변환 스킵 플래그를 포함하는
영상 정보를 수신
S810
현재 블록에 변환 스킵이
적용되었는지 여부를 나타내는
변환 스킵 플래그(transform skip flag)에 대한
컨텍스트 모델을 도출
S820
컨텍스트 모델을 기반으로
변환 스킵 플래그를 디코딩
S830
디코딩된 변환 스킵 플래그를 기반으로
레지듀얼 샘플을 도출
S840
레지듀얼 샘플을 기반으로 복원 픽처를 생성
S850

[도9]


310
330
디코딩 장치
비트스트림
엔트로피
디코딩부
예측부
레지듀얼
처리부
가산부
복원 픽처
320
340

[도10]


User Equipment
Game console
PC
Settop box
Smart Phone
...
Wired/Wireless communication
Wired/Wireless communication
Web Server
Streaming Server
Media storage
Realtime transmission
Encoding Server
Smart Phone
Camcoder/Camera
...

INTERNATIONAL SEARCH REPORT

International application No.
PCT/KR2020/003515

A. CLASSIFICATION OF SUBJECT MATTER
H04N 19/61(2014.01)i, H04N 19/11(2014.01)i, H04N 19/593(2014.01)i, H04N 19/132(2014.01)i, H04N 19/137(2014.01)i, H04N 19/70(2014.01)i, H04N 19/176(2014.01)i
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04N 19/61; H04N 19/00; H04N 19/13; H04N 19/176; H04N 19/593; H04N 19/70; H04N 19/91; H04N 19/11; H04N 19/132; H04N 19/137

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models: IPC as above
Japanese utility models and applications for utility models: IPC as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS (KIPO internal) & Keywords: transform, skip, residual, context, index, flag, ctxInc, ctxIdx

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	JP 2018-078647 A (SHARP CORP.) 17 May 2018 See paragraphs [0032], [0139], [0174].	1-2,8-9,15
A		3-7,10-14
Y	KR 10-2016-0072100 A (LG ELECTRONICS INC.) 22 June 2016 See paragraphs [0130], [0171].	1-2,8-9,15
Y	KR 10-2017-0046112 A (RESEARCH & BUSINESS FOUNDATION SUNGKYUNKWAN UNIVERSITY) 28 April 2017 See paragraph [0098].	2,9
A	US 2018-0324463 A1 (QUALCOMM INCORPORATED) 08 November 2018 See paragraph [0103].	1-15
A	KR 10-2014-0005101 A (SAMSUNG ELECTRONICS CO., LTD.) 14 January 2014 See claims 4-6.	1-15

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 JUNE 2020 (19.06.2020)	**22 JUNE 2020 (22.06.2020)**
Name and mailing address of the ISA/KR Korean Intellectual Property Office Government Complex Daejeon Building 4, 189, Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea Facsimile No. +82-42-481-8578	Authorized officer Telephone No.

Form PCT/ISA/210 (second sheet) (January 2015)

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/KR2020/003515

Patent document cited in search report	Publication date	Patent family member	Publication date
JP 2018-078647 A	17/05/2018	CN 104380737 A	25/02/2015
		EP 2866443 A1	29/04/2015
		US 2015-0181237 A1	25/06/2015
		US 2016-0134884 A1	12/05/2016
		US 2017-0099488 A1	06/04/2017
		US 9307264 B2	05/04/2016
		US 9516333 B2	06/12/2016
		US 9813713 B2	07/11/2017
		WO 2013-190990 A1	27/12/2013
KR 10-2016-0072100 A	22/06/2016	CN 105659605 A	08/06/2016
		EP 3059967 A1	24/08/2016
		JP 2016-537858 A	01/12/2016
		US 10321157 B2	11/06/2019
		US 2016-0249066 A1	25/08/2016
		WO 2015-057036 A1	23/04/2015
KR 10-2017-0046112 A	28/04/2017	KR 10-2016-0043496 A	21/04/2016
		KR 10-2020-0031591 A	24/03/2020
US 2018-0324463 A1	08/11/2018	US 2018-0324464 A1	08/11/2018
		WO 2018-208803 A1	15/11/2018
		WO 2018-208817 A1	15/11/2018
KR 10-2014-0005101 A	14/01/2014	CN 104811707 A	29/07/2015
		CN 104811708 A	29/07/2015
		CN 105007489 A	28/10/2015
		EP 3361732 A1	15/08/2018
		EP 3361733 A1	15/08/2018
		EP 3361734 A1	15/08/2018
		EP 3361735 A1	15/08/2018
		JP 2015-526014 A	07/09/2015
		JP 2016-213895 A	15/12/2016
		JP 2018-046573 A	22/03/2018
		KR 10-2014-0093201 A	25/07/2014
		KR 10-2014-0122698 A	20/10/2014
		US 2015-0117546 A1	30/04/2015
		US 2015-0139297 A1	21/05/2015
		US 2015-0189291 A1	02/07/2015
		US 2015-0195583 A1	09/07/2015
		US 2015-0195584 A1	09/07/2015
		WO 2014-007524 A1	09/01/2014

국제조사보고서

국제출원번호
PCT/KR2020/003515

A. 발명이 속하는 기술분류(국제특허분류(IPC))

H04N 19/61(2014.01)i, H04N 19/11(2014.01)i, H04N 19/593(2014.01)i, H04N 19/132(2014.01)i, H04N 19/137(2014.01)i, H04N 19/70(2014.01)i, H04N 19/176(2014.01)i

B. 조사된 분야

조사된 최소문헌(국제특허분류를 기재)
H04N 19/61; H04N 19/00; H04N 19/13; H04N 19/176; H04N 19/593; H04N 19/70; H04N 19/91; H04N 19/11; H04N 19/132; H04N 19/137

조사된 기술분야에 속하는 최소문헌 이외의 문헌
한국등록실용신안공보 및 한국공개실용신안공보: 조사된 최소문헌란에 기재된 IPC
일본등록실용신안공보 및 일본공개실용신안공보: 조사된 최소문헌란에 기재된 IPC

국제조사에 이용된 전산 데이터베이스(데이터베이스의 명칭 및 검색어(해당하는 경우))
eKOMPASS(특허청 내부 검색시스템) & 키워드: 변환(transform), 스킵(skip), 레지듀얼(residual), 콘텍스트(context), 인덱스(index), 플래그(flag), ctxInc, ctxIdx

C. 관련 문헌

카테고리*	인용문헌명 및 관련 구절(해당하는 경우)의 기재	관련 청구항
Y	JP 2018-078647 A (SHARP CORP.) 2018.05.17 단락 [0032], [0139], [0174]	1-2,8-9,15
A		3-7,10-14
Y	KR 10-2016-0072100 A (엘지전자 주식회사) 2016.06.22 단락 [0130], [0171]	1-2,8-9,15
Y	KR 10-2017-0046112 A (성균관대학교산학협력단) 2017.04.28 단락 [0098]	2,9
A	US 2018-0324463 A1 (QUALCOMM INCORPORATED) 2018.11.08 단락 [0103]	1-15
A	KR 10-2014-0005101 A (삼성전자주식회사) 2014.01.14 청구항 4-6	1-15

☐ 추가 문헌이 C(계속)에 기재되어 있습니다.

☒ 대응특허에 관한 별지를 참조하십시오.

* 인용된 문헌의 특별 카테고리:

"A" 특별히 관련이 없는 것으로 보이는 일반적인 기술수준을 정의한 문헌

"D" 본 국제출원에서 출원인이 인용한 문헌

"E" 국제출원일보다 빠른 출원일 또는 우선일을 가지나 국제출원일 이후에 공개된 선출원 또는 특허 문헌

"L" 우선권 주장에 의문을 제기하는 문헌 또는 다른 인용문헌의 공개일 또는 다른 특별한 이유(이유를 명시)를 밝히기 위하여 인용된 문헌

"O" 구두 개시, 사용, 전시 또는 기타 수단을 언급하고 있는 문헌

"P" 우선일 이후에 공개되었으나 국제출원일 이전에 공개된 문헌

"T" 국제출원일 또는 우선일 후에 공개된 문헌으로, 출원과 상충하지 않으며 발명의 기초가 되는 원리나 이론을 이해하기 위해 인용된 문헌

"X" 특별한 관련이 있는 문헌. 해당 문헌 하나만으로 청구된 발명의 신규성 또는 진보성이 없는 것으로 본다.

"Y" 특별한 관련이 있는 문헌. 해당 문헌이 하나 이상의 다른 문헌과 조합하는 경우로 그 조합이 당업자에게 자명한 경우 청구된 발명은 진보성이 없는 것으로 본다.

"&" 동일한 대응특허문헌에 속하는 문헌

국제조사의 실제 완료일	국제조사보고서 발송일
2020년 06월 19일 (19.06.2020)	2020년 06월 22일 (22.06.2020)

ISA/KR의 명칭 및 우편주소
대한민국 특허청
(35208) 대전광역시 서구 청사로 189,
4동 (둔산동, 정부대전청사)
팩스 번호 +82-42-481-8578

심사관
김성훈
전화번호 +82-42-481-8710

국제조사보고서
대응특허에 관한 정보

국제출원번호
PCT/KR2020/003515

국제조사보고서에서 인용된 특허문헌	공개일	대응특허문헌	공개일
JP 2018-078647 A	2018/05/17	CN 104380737 A	2015/02/25
		EP 2866443 A1	2015/04/29
		US 2015-0181237 A1	2015/06/25
		US 2016-0134884 A1	2016/05/12
		US 2017-0099488 A1	2017/04/06
		US 9307264 B2	2016/04/05
		US 9516333 B2	2016/12/06
		US 9813713 B2	2017/11/07
		WO 2013-190990 A1	2013/12/27
KR 10-2016-0072100 A	2016/06/22	CN 105659605 A	2016/06/08
		EP 3059967 A1	2016/08/24
		JP 2016-537858 A	2016/12/01
		US 10321157 B2	2019/06/11
		US 2016-0249066 A1	2016/08/25
		WO 2015-057036 A1	2015/04/23
KR 10-2017-0046112 A	2017/04/28	KR 10-2016-0043496 A	2016/04/21
		KR 10-2020-0031591 A	2020/03/24
US 2018-0324463 A1	2018/11/08	US 2018-0324464 A1	2018/11/08
		WO 2018-208803 A1	2018/11/15
		WO 2018-208817 A1	2018/11/15
KR 10-2014-0005101 A	2014/01/14	CN 104811707 A	2015/07/29
		CN 104811708 A	2015/07/29
		CN 105007489 A	2015/10/28
		EP 3361732 A1	2018/08/15
		EP 3361733 A1	2018/08/15
		EP 3361734 A1	2018/08/15
		EP 3361735 A1	2018/08/15
		JP 2015-526014 A	2015/09/07
		JP 2016-213895 A	2016/12/15
		JP 2018-046573 A	2018/03/22
		KR 10-2014-0093201 A	2014/07/25
		KR 10-2014-0122698 A	2014/10/20
		US 2015-0117546 A1	2015/04/30
		US 2015-0139297 A1	2015/05/21
		US 2015-0189291 A1	2015/07/02
		US 2015-0195583 A1	2015/07/09
		US 2015-0195584 A1	2015/07/09
		WO 2014-007524 A1	2014/01/09