(12) 특허협력조약에 의하여 공개된 국제출원

(19) 세계지식재산권기구
국제사무국

(43) 국제공개일
2018 년 11 월 15 일 (15.11.2018)

WIPO | PCT

(10) 국제공개번호
WO 2018/207956 A1

(51) 국제특허분류:
H04N 19/13 (2014.01) *H04N 19/176* (2014.01)
H04N 19/70 (2014.01) *H04N 19/68* (2014.01)

(21) 국제출원번호: PCT/KR2017/004818

(22) 국제출원일: 2017 년 5 월 10 일 (10.05.2017)

(25) 출원언어: 한국어

(26) 공개언어: 한국어

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

(72) 발명자: 구문모 **(KOO, Moonmo)**; 06772 서울시 서초구 양재대로11길 19, LG전자 특허센터, Seoul (KR).

(74) 대리인: 특허법인 로얄 **(ROYAL PATENT & LAW OFFICE)**; 08806 서울시 관악구 남부순환로 2072, 도원회관 빌딩 1층, 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, 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, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

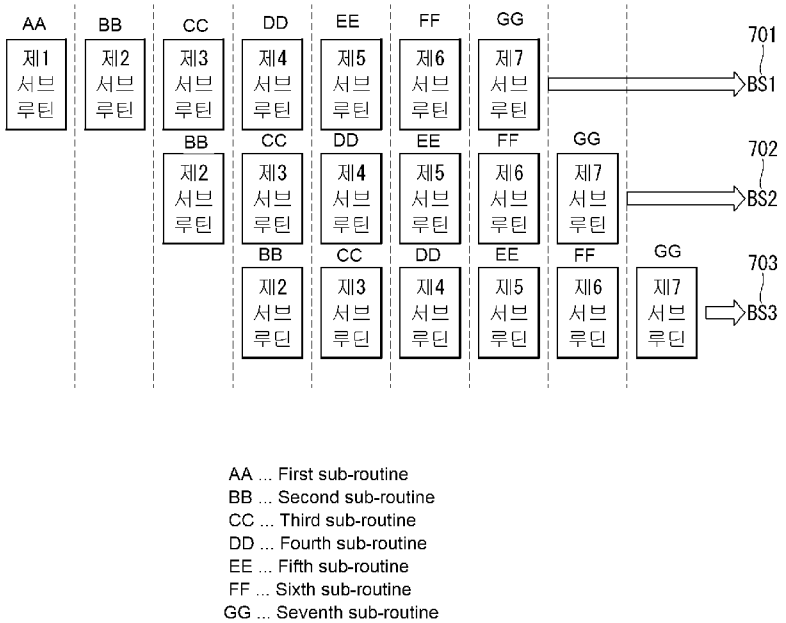
(84) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 역내 권리의 보호를 위하여): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 유라시아 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 유럽 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS,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

WO 2018/207956 A1

(54) **Title:** METHOD AND DEVICE FOR ENTROPY-ENCODING AND ENTROPY-DECODING VIDEO SIGNAL

(54) **발명의 명칭:** 비디오 신호를 엔트로피 인코딩, 디코딩하는 방법 및 장치

[도7]



AA ... First sub-routine
BB ... Second sub-routine
CC ... Third sub-routine
DD ... Fourth sub-routine
EE ... Fifth sub-routine
FF ... Sixth sub-routine
GG ... Seventh sub-routine

(57) **Abstract:** The present invention provides a method for performing entropy-decoding a video signal, the method comprising the steps of: receiving a video signal including a plurality of bit streams; decoding a symbol of a syntax element for each bit stream of the video signal; and deriving information of the syntax element by using the decoded symbol, wherein the syntax element is alternately mapped to the plurality of bit streams on the basis of a function block unit indicating an area in which one or more syntax elements are independently processed within a current block.

[다음 쪽 계속]

(BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

공개:

— 국제조사보고서와 함께 (조약 제21조(3))

(57) 요약서: 본 발명은, 비디오 신호에 대해 엔트로피 디코딩을 수행하는 방법에 있어서, 복수 개의 비트 스트림(bit stream)으로 구성된 비디오 신호를 수신하는 단계; 상기 비디오 신호의 각 비트 스트림 별로 신택스 요소(syntax element)의 심볼(symbol)을 복호화하는 단계; 및 상기 복호화된 심볼을 이용하여 상기 신택스 요소의 정보를 도출하는 단계를 포함하되, 상기 신택스 요소는 현재 블록 내에서 하나 이상의 신택스 요소가 독립적으로 처리되는 영역을 나타내는 기능 블록 단위(function block unit)로 상기 복수 개의 비트 스트림에 번갈아가며 매핑되는 방법을 제공한다.

명세서

발명의 명칭: 비디오 신호를 엔트로피 인코딩, 디코딩하는 방법 및 장치

기술분야

[1] 본 발명은 비디오 신호를 엔트로피 인코딩, 디코딩하는 방법 및 장치에 관한 것이다. 보다 구체적으로, 여러 비트 스트림을 이용하여 비디오 신호를 엔트로피 인코딩, 디코딩하는 방법 및 장치에 관한 것이다.

배경기술

[2] 엔트로피 코딩은 부호화 과정을 통해 결정된 신택스 엘리먼트들을 무손실 압축하여 RBSP(Raw Byte Sequence Payload)를 생성하는 과정이다. 엔트로피 코딩은 신택스의 통계를 이용하여 자주 발생하는 신택스에 대해서는 짧은 비트를 할당하고, 그렇지 않은 신택스에는 긴 비트를 할당하여 신택스 엘리먼트들을 간결한 데이터로 표현한다.

[3] 그 중, CABAC(Context-based Adaptive Binary Arithmetic Coding)은 이진 산술 코딩을 수행하는 과정에서 신택스의 컨텍스트와 이전에 발생한 심볼에 기초하여 적응적으로 업데이트된 컨텍스트 모델을 사용한다. 그러나, 이러한 CABAC도 연산량이 많아 복잡도가 높고 순차적 구조를 가지고 있어 병렬 수행이 어려운 문제점이 있다.

[4] 따라서, 비디오 압축 기술에 있어서 신택스 엘리먼트를 보다 효율적으로 압축하고 전송할 필요가 있으며, 이를 위해 엔트로피 코딩의 성능을 향상시킬 필요가 있다.

발명의 상세한 설명

기술적 과제

[5] 본 발명의 목적은 엔트로피 코딩의 병렬화를 위해 신택스 요소(syntax element)를 기능 블록 단위(function block unit)로 여러 비트 스트림(bit stream)에 매핑하여 코딩하는 방법을 제안한다.

[6] 또한, 본 발명의 목적은 기능 블록 단위 또는 신택스 요소 단위로 여러 비트 스트림간 동기화를 수행하는 방법을 제안한다.

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

과제 해결 수단

[8] 본 발명의 일 양상은, 비디오 신호에 대해 엔트로피 디코딩을 수행하는 방법에 있어서, 복수 개의 비트 스트림(bit stream)으로 구성된 비디오 신호를 수신하는 단계; 상기 비디오 신호의 각 비트 스트림 별로 신택스 요소(syntax element)의

심볼(symbol)을 복호화하는 단계; 및 상기 복호화된 심볼을 이용하여 상기 신택스 요소의 정보를 도출하는 단계를 포함하되, 상기 신택스 요소는 현재 블록 내에서 하나 이상의 신택스 요소가 독립적으로 처리되는 영역을 나타내는 기능 블록 단위(function block unit)로 상기 복수 개의 비트 스트림에 번갈아가며 매핑될 수 있다.

[9] 바람직하게, 상기 신택스 요소의 심볼을 복호화하는 단계는, 상기 기능 블록 단위로 상기 복수 개의 비트 스트림간 동기화를 수행하는 단계를 더 포함할 수 있다.

[10] 바람직하게, 상기 신택스 요소의 심볼을 복호화하는 단계는, 상기 신택스 요소 단위로 상기 복수 개의 비트 스트림간 동기화를 수행하는 단계를 더 포함할 수 있다.

[11] 바람직하게, 상기 복수 개의 비트 스트림간 동기화를 수행하는 단계는, 현재 비트 스트림에서 현재 신택스 요소의 복호화를 시작하기 위한 락(lock)을 획득하는 단계를 더 포함하고, 상기 락을 획득한 경우에만 상기 현재 신택스 요소의 심볼을 복호화할 수 있다.

[12] 바람직하게, 상기 복수 개의 비트 스트림간 동기화를 수행하는 단계는, 상기 현재 신택스 요소의 복호화가 종료된 경우 상기 획득된 락을 반환하는 단계를 더 포함할 수 있다.

[13] 바람직하게, 상기 복수 개의 비트 스트림간 동기화를 수행하는 단계는, 상기 현재 비트 스트림 이외의 비트 스트림에서 상기 현재 신택스 요소에 대한 락을 획득하였는지 여부를 나타내는 락 상태 정보(lock state information)를 확인하는 단계를 더 포함할 수 있다.

[14] 바람직하게, 상기 복수 개의 비트 스트림간 동기화를 수행하는 단계는, 현재 신택스 요소에 컨텍스트(context)를 참조하는 정규 코딩(regular coding)이 적용되는 경우, 상기 복수 개의 비트 스트림간 동기화를 수행할 수 있다.

[15] 바람직하게, 상기 신택스 요소의 심볼을 복호화하는 단계는, 상기 기능 블록 단위로 신택스 요소의 심볼을 복호화한 후 현재 비트 스트림을 전환하는 단계를 더 포함할 수 있다.

[16] 바람직하게, 상기 기능 블록 단위 마다 독립적으로 처리되는 작업 단위를 나타내는 스레드(thread)를 생성하여 상기 스레드의 정보를 저장하는 스레드 큐(thread queue)에 등록하는 단계; 및 상기 스레드 큐에 저장된 복수 개의 스레드들을 병렬적으로 복호화하는 단계를 더 포함할 수 있다.

[17] 본 발명의 다른 일 양상은, 비디오 신호에 대해 엔트로피 디코딩을 수행하는 장치에 있어서, 복수 개의 비트 스트림(bit stream)으로 구성된 비디오 신호를 수신하는 비디오 신호 수신부; 상기 비디오 신호의 각 비트 스트림 별로 신택스 요소(syntax element)의 심볼(symbol)을 복호화하는 심볼 복호화부; 및 상기 복호화된 심볼을 이용하여 상기 신택스 요소의 정보를 복호화하는 신택스 요소 도출부를 포함하되, 상기 신택스 요소는 현재 블록 내에서 하나 이상의 신택스

요소가 독립적으로 처리되는 영역을 나타내는 기능 블록 단위(function block unit)로 상기 복수 개의 비트 스트림에 번갈아가며 매핑될 수 있다.

[18] 바람직하게, 상기 심볼 복호화부는, 상기 기능 블록 단위로 상기 복수 개의 비트 스트림간 동기화를 수행하는 동기화부를 포함할 수 있다.

[19] 바람직하게, 상기 심볼 복호화부는, 상기 신택스 요소 단위로 상기 복수 개의 비트 스트림간 동기화를 수행하는 동기화부를 포함할 수 있다.

[20] 바람직하게, 상기 동기화부는, 현재 비트 스트림에서 현재 신택스 요소의 복호화를 시작하기 위한 락(lock)을 획득하고, 상기 락이 획득된 경우에만 상기 현재 신택스 요소의 심볼이 복호화될 수 있다.

[21] 바람직하게, 상기 동기화부는, 상기 현재 신택스 요소의 복호화가 종료된 경우 상기 획득된 락을 반환할 수 있다.

[22] 바람직하게, 상기 동기화부는, 상기 현재 비트 스트림 이외의 비트 스트림에서 상기 현재 신택스 요소에 대한 락을 획득하였는지 여부를 나타내는 락 상태 정보(lock state information)를 확인할 수 있다.

[23] 바람직하게, 상기 동기화부는, 현재 신택스 요소에 컨텍스트(context)를 참조하는 정규 코딩(regular coding)이 적용되는 경우, 상기 복수 개의 비트 스트림간 동기화를 수행할 수 있다.

[24] 바람직하게, 상기 심볼 복호화부는, 상기 기능 블록 단위로 신택스 요소의 심볼을 복호화한 후 현재 비트 스트림을 전환할 수 있다.

[25] 바람직하게, 상기 기능 블록 단위 마다 독립적으로 처리되는 작업 단위를 나타내는 스레드(thread)를 생성하여 상기 스레드의 정보를 저장하는 스레드 큐(thread queue)에 등록하는 스레드 생성부; 및 상기 스레드 큐에 저장된 복수 개의 스레드들을 병렬적으로 복호화하는 스레드 복호화부를 더 포함할 수 있다.

발명의 효과

[26] 본 발명의 실시예에 따르면, 신택스 요소(syntax element)들을 기능 블록 단위(function block unit)로 여러 비트 스트림에 코딩하여 병렬화함으로써 엔트로피 코딩의 처리량(throughput)을 향상시킬 수 있다.

[27] 본 발명의 실시예에 따르면, 여러 비트 스트림들간 동기화를 수행함으로써, 코딩 순서를 지키면서 신택스 요소들을 병렬적으로 코딩할 수 있다.

[28] 또한, 본 발명의 실시예에 따르면, 정규 코딩(regular coding)을 적용한 심볼들과, 바이패스 코딩(bypass coding)을 적용한 심볼들을 여러 비트 스트림에 할당하여 병렬적으로 코딩함으로써 처리량을 향상시키고고 복잡도를 개선할 수 있다.

[29] 또한, 본 발명의 실시예에 따르면, 데이터 의존성(data dependency)이 최소화되도록 심볼 그룹을 구성함으로써 엔트로피 코딩의 처리량을 향상시킬 수 있다.

[30] 본 발명에서 얻을 수 있는 효과는 이상에서 언급한 효과로 제한되지 않으며, 언급하지 않은 또 다른 효과들은 아래의 기재로부터 본 발명이 속하는

기술분야에서 통상의 지식을 가진 자에게 명확하게 이해될 수 있을 것이다.

도면의 간단한 설명

[31] 도 1은 본 발명이 적용되는 실시예로서, 비디오 신호의 인코딩이 수행되는 인코더의 개략적인 블록도를 나타낸다.

[32] 도 2는 본 발명이 적용되는 실시예로서, 비디오 신호의 디코딩이 수행되는 디코더의 개략적인 블록도를 나타낸다.

[33] 도 3은 본 발명이 적용되는 실시예로서, CABAC(Context-based Adaptive Binary Arithmetic Coding)이 적용되는 엔트로피 인코딩부의 개략적인 블록도를 나타낸다.

[34] 도 4는 본 발명이 적용되는 실시예로서, CABAC(Context-based Adaptive Binary Arithmetic Coding)이 적용되는 엔트로피 디코딩부의 개략적인 블록도를 나타낸다.

[35] 도 5는 본 발명이 적용되는 실시예로서, CABAC(Context-based Adaptive Binary Arithmetic Coding)에 따라 수행되는 인코딩 흐름도를 나타낸다.

[36] 도 6은 본 발명이 적용되는 실시예로서, CABAC(Context-based Adaptive Binary Arithmetic Coding)에 따라 수행되는 디코딩 흐름도를 나타낸다.

[37] 도 7은 본 발명이 적용될 수 있는 실시예로서, 레지듀얼 코딩 신택스(residual coding syntax)의 신택스 요소(syntax element)들을 기능 블록 단위(function block unit)로 여러 비트 스트림에 매핑하는 방법을 예시하는 도면이다.

[38] 도 8은 본 발명이 적용될 수 있는 실시예로서, 신택스 요소(syntax element) 단위로 여러 비트 스트림간 동기화를 수행하는 방법을 예시하는 레지듀얼 코딩(residual coding) 신택스이다.

[39] 도 9는 본 발명이 적용되는 일 실시예로서, 레지듀얼 코딩(residual coding)의 신택스 요소(syntax element)를 기능 블록 단위(function block unit)로 여러 비트 스트림에서 코딩하는 방법을 예시하는 신택스이다.

[40] 도 10은 본 발명이 적용될 수 있는 실시예로서, 코딩 유닛(coding unit)의 신택스 요소를 여러 비트 스트림에서 코딩하는 방법을 예시하는 신택스이다.

[41] 도 11은 본 발명이 적용될 수 있는 실시예로서, 예측 유닛(prediction unit)의 신택스 요소를 여러 비트 스트림에서 코딩하는 방법을 예시하는 신택스이다.

[42] 도 12는 본 발명이 적용될 수 있는 실시예로서, 예측 유닛(prediction unit)의 신택스 요소를 여러 비트 스트림에서 코딩하는 방법을 예시하는 신택스이다.

[43] 도 13은 본 발명이 적용될 수 있는 실시예로서, 코딩 유닛(coding unit)의 신택스 요소를 여러 비트 스트림에서 코딩하는 방법을 예시하는 도면이다.

[44] 도 14는 본 발명이 적용될 수 있는 실시예로서, 신택스 요소(syntax element) 단위로 여러 비트 스트림간 동기화를 수행하는 방법을 예시하는 코딩 유닛(coding unit) 신택스이다.

[45] 도 15는 본 발명이 적용될 수 있는 실시예로서, 예측 유닛(prediction unit)의

신택스 요소를 여러 비트 스트림에서 코딩하는 방법을 예시하는 신택스이다.

[46] 도 16는 본 발명이 적용될 수 있는 실시예로서, PU(prediction unit)의 신택스 요소를 여러 비트 스트림에서 코딩하는 방법을 예시하는 도면이다.

[47] 도 17은 본 발명이 적용될 수 있는 실시예로서, 신택스 요소(syntax element) 단위로 여러 비트 스트림간 동기화를 수행하는 방법을 예시하는 예측 유닛(coding unit) 신택스이다.

[48] 도 18은 본 발명이 적용될 수 있는 실시예로서, 코딩 트리 유닛(CTU)으로부터 계층적으로 분할되는 유닛들을 여러 비트 스트림에 매핑하여 코딩하는 방법을 설명하기 위한 도면이다.

[49] 도 19는 본 발명이 적용될 수 있는 실시예로서, 여러 비트 스트림으로 구성된 비디오 신호에 대해 엔트로피 디코딩을 수행하는 과정을 설명하는 흐름도이다.

발명의 실시를 위한 형태

[50] 이하, 본 발명의 실시예들에 따른 예시적인 엘리먼트들 및 동작들이 첨부된 도면들을 참조하여 기술된다. 그러나 상기 도면들을 참조하여 기술되는 본 발명의 상기 엘리먼트들 및 동작들은 단지 실시예들로서 제공되는 것이고, 이것에 의해서 본 발명의 기술적 사상과 그 핵심 구성 및 작용이 제한되지는 않는다는 것을 밝혀두고자 한다.

[51] 또한, 본 발명에서 사용되는 용어는 가능한 한 현재 널리 사용되는 일반적인 용어를 선택하였으나, 특정한 경우는 출원인이 임의로 선정한 용어를 사용하여 설명한다. 그러한 경우에는 해당 부분의 상세 설명에서 그 의미가 명확하게 기재된다. 따라서, 본 명세서의 설명에서 사용된 용어의 명칭 만에 기초하여 본 발명이 단순 해석되어서는 안 될 것이며 그 해당 용어의 의미까지 파악하여 해석되어야 함을 밝혀두고자 한다.

[52] 또한, 본 발명에서 사용되는 용어들은 발명을 설명하기 위해 선택된 일반적인 용어들이나, 유사한 의미를 갖는 다른 용어가 있는 경우 보다 적절한 해석을 위해 대체 가능할 것이다. 예를 들어, 신호, 데이터, 샘플, 픽쳐, 프레임, 블록은 각 코딩 과정에서 적절하게 대체되어 해석될 수 있을 것이다.

[53] 또한, 본 명세서에서 설명되는 실시예들의 개념과 방법들은, 다른 실시예들에도 적용가능하며, 본 명세서에서 모두 명시하여 기재하지 않더라도 본 발명의 기술적 사상 범위 내에서 각 실시예들의 조합도 적용가능할 것이다.

[54]

[55] 도 1은 본 발명이 적용되는 실시예로서, 비디오 신호의 인코딩이 수행되는 인코더의 개략적인 블록도를 나타낸다.

[56] 도 1을 참조하면, 인코더(100)는 영상 분할부(110), 변환부(120), 양자화부(130), 역양자화부(140), 역변환부(150), 필터링부(160), 복호 픽쳐 버퍼(DPB: Decoded Picture Buffer)(170), 인터 예측부(180), 인트라 예측부(185) 및 엔트로피 인코딩부(190)를 포함하여 구성될 수 있다.

[57] 영상 분할부(110)는 인코더(100)에 입력된 입력 영상(또는, 픽쳐, 프레임)를 하나 이상의 처리 유닛으로 분할할 수 있다. 예를 들어, 상기 처리 유닛은 코딩 트리 유닛(CTU: Coding Tree Unit), 코딩 유닛(CU: Coding Unit), 예측 유닛(PU: Prediction Unit) 또는 변환 유닛(TU: Transform Unit)일 수 있다.

[58] 인코더(100)는 입력 영상 신호에서 인터 예측부(180) 또는 인트라 예측부(185)로부터 출력된 예측 신호를 감산하여 잔여 신호(residual signal)를 생성할 수 있고, 생성된 잔여 신호는 변환부(120)로 전송된다.

[59] 변환부(120)는 잔여 신호에 변환 기법을 적용하여 변환 계수(transform coefficient)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.

[60] 양자화부(130)는 변환 계수를 양자화하여 엔트로피 인코딩부(190)로 전송하고, 엔트로피 인코딩부(190)는 양자화된 신호를 엔트로피 코딩하여 비트스트림으로 출력할 수 있다.

[61] 양자화부(130)로부터 출력된 양자화된 신호는 예측 신호를 생성하기 위해 이용될 수 있다. 예를 들어, 양자화된 신호는 루프 내의 역양자화부(140) 및 역변환부(150)를 통해 역양자화 및 역변환을 적용함으로써 잔여 신호를 복원할 수 있다. 복원된 잔여 신호를 인터 예측부(180) 또는 인트라 예측부(185)로부터 출력된 예측 신호에 더함으로써 복원 신호가 생성될 수 있다.

[62] 필터링부(160)는 복원 신호에 필터링을 적용하여 이를 재생 장치로 출력하거나 복호 픽쳐 버퍼(170)에 전송한다. 복호 픽쳐 버퍼(170)에 전송된 필터링된 신호는 인터 예측부(180)에서 참조 픽쳐로 사용될 수 있다. 이처럼, 필터링된 픽쳐를 화면간 예측 모드에서 참조 픽쳐로 이용함으로써 화질 뿐만 아니라 부호화 효율도 향상시킬 수 있다.

[63] 복호 픽쳐 버퍼(170)는 필터링된 픽쳐를 인터 예측부(180)에서의 참조 픽쳐로 사용하기 위해 저장할 수 있다.

[64] 인터 예측부(180)는 복원 픽쳐를 참조하여 시간적 중복성 및/또는 공간적 중복성을 제거하기 위해 시간적 예측 및/또는 공간적 예측을 수행한다. 이때, 인터 예측 모드에서 전송되는 움직임 정보의 양을 줄이기 위해 주변 블록과 현재 블록 간의 움직임 정보의 상관성에 기초하여 움직임 정보를 예측할 수 있다.

[65] 인트라 예측부(185)는 현재 부호화를 진행하려고 하는 블록의 주변에 있는 샘플들을 참조하여 현재 블록을 예측할 수 있다. 상기 인트라 예측부(185)는

인트라 예측을 수행하기 위해 다음과 같은 과정을 수행할 수 있다. 먼저, 예측 신호를 생성하기 위해 필요한 참조 샘플을 준비할 수 있다. 그리고, 준비된 참조 샘플을 이용하여 예측 신호를 생성할 수 있다. 이후, 예측 모드를 부호화하게 된다. 이때, 참조 샘플은 참조 샘플 패딩 및/또는 참조 샘플 필터링을 통해 준비될 수 있다. 참조 샘플은 예측 및 복원 과정을 거쳤기 때문에 양자화 에러가 존재할 수 있다. 따라서, 이러한 에러를 줄이기 위해 인트라 예측에 이용되는 각 예측 모드에 대해 참조 샘플 필터링 과정이 수행될 수 있다.

[66] 상기 인터 예측부(180) 또는 상기 인트라 예측부(185)를 통해 생성된 예측 신호는 복원 신호를 생성하기 위해 이용되거나 잔여 신호를 생성하기 위해 이용될 수 있다.

[67]

[68] 도 2는 본 발명이 적용되는 실시예로서, 비디오 신호의 디코딩이 수행되는 디코더의 개략적인 블록도를 나타낸다.

[69] 도 2를 참조하면, 디코더(200)는 엔트로피 디코딩부(210), 역양자화부(220), 역변환부(230), 필터링부(240), 복호 픽쳐 버퍼(DPB: Decoded Picture Buffer Unit)(250), 인터 예측부(260) 및 인트라 예측부(265)를 포함하여 구성될 수 있다.

[70] 그리고, 디코더(200)를 통해 출력된 복원 영상 신호는 재생 장치를 통해 재생될 수 있다.

[71] 디코더(200)는 도 1의 인코더(100)로부터 출력된 신호을 수신할 수 있고, 수신된 신호는 엔트로피 디코딩부(210)를 통해 엔트로피 디코딩될 수 있다.

[72] 역양자화부(220)에서는 양자화 스텝 사이즈 정보를 이용하여 엔트로피 디코딩된 신호로부터 변환 계수(transform coefficient)를 획득한다.

[73] 역변환부(230)에서는 변환 계수를 역변환하여 잔여 신호를 획득하게 된다.

[74] 획득된 잔여 신호를 인터 예측부(260) 또는 인트라 예측부(265)로부터 출력된 예측 신호에 더함으로써 복원 신호가 생성된다.

[75] 필터링부(240)는 복원 신호에 필터링을 적용하여 이를 재생 장치로 출력하거나 복호 픽쳐 버퍼부(250)에 전송한다. 복호 픽쳐 버퍼부(250)에 전송된 필터링된 신호는 인터 예측부(260)에서 참조 픽쳐로 사용될 수 있다.

[76] 본 명세서에서, 인코더(100)의 필터링부(160), 인터 예측부(180) 및 인트라 예측부(185)에서 설명된 실시예들은 각각 디코더의 필터링부(240), 인터 예측부(260) 및 인트라 예측부(265)에도 동일하게 적용될 수 있다.

[77]

[78] 도 3은 본 발명이 적용되는 실시예로서, CABAC(Context-based Adaptive Binary Arithmetic Coding)이 적용되는 엔트로피 인코딩부의 개략적인 블록도를 나타낸다.

[79] 본 발명이 적용되는 엔트로피 인코딩부(300)는 이진화부(310), 컨텍스트 모델링부(320), 이진 산술 인코딩부(330) 및 메모리(360)를 포함하고, 상기 이진 산술 인코딩부(330)는 정규 이진 인코딩부(regular binary encoding unit)(340) 및

바이패스 이진 인코딩부(bypass binary encoding unit)(350)를 포함한다. 여기서, 상기 정규 이진 인코딩부(regular binary encoding unit)(340) 및 상기 바이패스 이진 인코딩부(bypass binary encoding unit)(350)는 각각 정규 코딩 엔진, 바이패스 코딩 엔진이라 불릴 수 있다.

[80] 상기 이진화부(310)은 데이터 심볼들의 시퀀스를 수신하고 이진화를 수행함으로써 0 또는 1의 이진화된 값으로 구성된 이진 심볼(bin) 스트링을 출력할 수 있다. 상기 이진화부(310)은 신택스(syntax) 요소들을 이진 심볼들로 매핑할 수 있다. 서로 다른 여러 이진화 과정들, 예를 들어, 단항(unary: U), 끝이 잘린 단항(truncated unary: TU), k차 Exp-Golomb (EGk), 및 고정 길이(Fixed Length) 과정 등이 이진화를 위해 사용될 수 있다. 상기 이진화 과정은 신택스 요소의 유형을 기반으로 선택될 수 있다.

[81] 출력된 이진 심볼 스트링은 컨텍스트 모델링부(320)으로 전송된다.

[82] 상기 컨텍스트 모델링부(320)은 메모리로부터 현재 블록을 코딩하는데 필요한 확률 정보를 선택하여 상기 이진 산술 인코딩부(330)로 전송한다. 예를 들어, 코딩할 신택스 엘리먼트에 기초하여 상기 메모리(360)에서 컨텍스트 메모리를 선택하고 빈 인덱스(binIdx)를 통해 현재 신택스 엘리먼트 코딩에 필요한 확률 정보를 선택할 수 있다. 여기서, 컨텍스트는 심볼의 발생 확률에 관한 정보를 의미하고, 컨텍스트 모델링은 이전까지 코딩된 빈(bin)들에 대한 정보로부터 다음 빈(bin)의 이진 산술 코딩에 필요한 확률을 추정하는 과정을 의미한다. 그리고, 컨텍스트는 특정 확률 값을 나타내는 상태(state)와 MPS(Most Probable Symbol)로 구성될 수 있다.

[83] 상기 컨텍스트 모델링부(320)는 높은 코딩 효율을 성취하기 위해 필요한 정확한 확률 추정을 제공할 수 있다. 이에 따라, 서로 다른 이진 심볼들에 대해 서로 다른 컨텍스트 모델들이 사용될 수 있고 이러한 컨텍스트 모델의 확률은 이전에 코딩된 이진 심볼의 값들을 기반으로 업데이트될 수 있다.

[84] 유사한 분포를 가지는 이진 심볼들은 동일한 컨텍스트 모델을 공유할 수 있다. 이러한 각 이진 심볼에 대한 컨텍스트 모델은 확률 추정을 위해, 빈의 신택스 정보, 빈 스트링에서의 빈의 위치를 나타내는 빈 인덱스(binIdx), 빈이 들어 있는 블록의 이웃 블록에 포함된 빈의 확률, 이웃 블록의 특정 신택스 요소의 디코딩 값 중 적어도 하나가 이용될 수 있다.

[85] 상기 이진 산술 인코딩부(330)는 정규 이진 인코딩부(regular binary encoding unit)(340) 및 바이패스 이진 인코딩부(bypass binary encoding unit)(350)를 포함하고, 출력된 스트링에 대한 엔트로피 인코딩을 수행하고 압축된 데이터 비트들을 출력한다.

[86] 상기 정규 이진 인코딩부(regular binary encoding unit)(340)는 재귀적 간격 분할(recursive interval division)을 기반으로 산술 코딩을 수행한다.

[87] 먼저, 0 내지 1의 초기값을 가지는 간격(또는 구간, 범위)이 이진 심볼의 확률을 기반으로 두 개의 하위 간격들로 분할된다. 인코딩된 비트들은 이진 소수로

변환되는 경우 이진 심볼의 값들이 연속적으로 디코딩되는 과정에서 0 또는 1을 나타내는 간격 중 하나를 선택할 수 있는 오프셋을 제공한다.

[88] 디코딩된 모드의 이진 심볼 이후에, 상기 간격은 선택된 하위 간격을 동일하게 하기 위해 업데이트될 수 있으며, 상기 간격 분할 과정 자체가 반복된다. 상기 간격 및 오프셋은 제한된 비트 정밀도를 가지며, 따라서 상기 간격이 특정한 값 아래로 떨어질 때마다 오버플로우를 방지하기 위해 재정규화(renormalization)가 필요할 수 있다. 상기 재정규화(renormalization)는 각각의 이진 심볼이 인코딩 또는 디코딩된 이후에 발생할 수 있다.

[89] 상기 바이패스 이진 인코딩부(bypass binary encoding unit)(350)는 컨텍스트 모델 없이 인코딩을 수행하며, 현재 코딩되는 빈(bin)의 확률을 0.5로 고정하여 코딩을 수행한다. 이는 신택스의 확률을 결정하기 어렵거나 고속으로 코딩하고자 할 때 이용될 수 있다.

[90]

[91] 도 4는 본 발명이 적용되는 실시예로서, CABAC(Context-based Adaptive Binary Arithmetic Coding)이 적용되는 엔트로피 디코딩부의 개략적인 블록도를 나타낸다.

[92] 본 발명이 적용되는 엔트로피 디코딩부(400)는 컨텍스트 모델링부(410), 이진 산술 디코딩부(420), 메모리(450) 및 역이진화부(460)를 포함하고, 상기 이진 산술 디코딩부(420)는 정규 이진 디코딩부(regular binary decoding unit)(430) 및 바이패스 이진 디코딩부(bypass binary decoding unit)(440)를 포함한다.

[93] 상기 엔트로피 디코딩부(400)는 비트 스트림을 수신하고 현재 신택스 요소에 바이패스 모드(bypass mode)가 적용되는지 여부를 확인한다. 여기서, 바이패스 모드(bypass mode)는 컨텍스트 모델을 이용하지 않고, 현재 코딩되는 빈(bin)의 확률을 0.5로 고정하여 코딩을 수행하는 것을 의미한다. 바이패스 모드(bypass mode)가 적용되지 않는 경우, 상기 정규 이진 디코딩부(regular binary decoding unit)(430)는 정규 모드(regular mode)에 따라 이진 산술 디코딩을 수행한다.

[94] 이때, 상기 컨텍스트 모델링부(410)는 상기 메모리(450)로부터 현재 비트스트림을 디코딩하는데 필요한 확률 정보를 선택하여 상기 정규 이진 디코딩부(regular binary decoding unit)(430)로 전송한다.

[95] 한편, 바이패스 모드(bypass mode)가 적용되는 경우, 상기 바이패스 이진 디코딩부(bypass binary decoding unit)(440)는 바이패스 모드(bypass mode)에 따라 이진 산술 디코딩을 수행한다.

[96] 상기 역이진화부(460)는 상기 이진 산술 디코딩부(420)에서 디코딩된 이진수 형태의 빈(bin)을 입력받아 정수 형태의 신택스 엘리먼트 값으로 변환 출력하게 된다. 반면, 이진수 형태의 빈(bin) 또는 빈 스트링(bin string)이 신택스 요소의 값으로 매핑되는 신택스 엘리먼트의 경우, 상기 역이진화부(460)는 상기 이진수 형태의 빈(bin)을 그대로 출력할 수도 있다,

[97]

[98] 도 5는 본 발명이 적용되는 실시예로서, CABAC(Context-based Adaptive Binary Arithmetic Coding)에 따라 수행되는 인코딩 흐름도를 나타낸다.

[99] 인코더는, 신택스 엘리먼트에 대해 이진화를 수행할 수 있다(S510).

[100] 상기 인코더는, 정규 모드에 따라 이진 산술 코딩을 수행할지 또는 바이패스 모드에 따라 이진 산술 코딩을 수행할지 여부를 확인할 수 있다(S520).

[101] 정규 모드의 경우, 상기 인코더는 컨텍스트 모델을 선택할 수 있고(S530), 상기 컨텍스트 모델에 기초하여 이진 산술 인코딩을 수행할 수 있다(S540). 그리고, 상기 인코더는, 컨텍스트 모델을 업데이트할 수 있으며(S550), 상기 S530 단계에서 업데이트된 컨텍스트 모델에 기초하여 다시 적합한 컨텍스트 모델을 선택할 수 있다.

[102] 한편, 바이패스 모드의 경우, 상기 인코더는 확률 0.5에 기초하여 이진 산술 인코딩을 수행할 수 있다(S560).

[103]

[104] 도 6은 본 발명이 적용되는 실시예로서, CABAC(Context-based Adaptive Binary Arithmetic Coding)에 따라 수행되는 디코딩 흐름도를 나타낸다.

[105] 먼저, 디코더는 비트스트림을 수신할 수 있다(S610).

[106] 상기 디코더는 현재 신택스 요소에 정규 모드(regular mode)가 적용되는지 또는 바이패스 모드(bypass mode)가 적용되는지 여부를 확인할 수 있다(S620). 여기서, 상기 바이패스 모드의 적용 여부는 신택스의 종류에 따라 사전에 결정되어 있을 수 있다.

[107] 또한, 정규 모드가 적용되는 심볼들과 바이패스 모드가 적용되는 심볼들이 조합되어 신택스 요소를 구성할 수도 있다. 이 경우, 상기 디코더는 현재 신택스 요소의 심볼들에 바이패스 모드(bypass mode)가 적용되는지 여부를 확인할 수 있다.

[108] 상기 S620단계에서 확인한 결과 정규 모드가 적용되는 경우, 상기 디코더는 컨텍스트 모델을 선택할 수 있고(S630), 상기 컨텍스트 모델에 기초하여 이진 산술 디코딩을 수행할 수 있다(S640). 그리고, 상기 디코더는, 컨텍스트 모델을 업데이트할 수 있으며(S650), 상기 S630 단계에서 업데이트된 컨텍스트 모델에 기초하여 다시 적합한 컨텍스트 모델을 선택할 수 있다.

[109] 한편, 상기 S620단계에서 확인한 결과 바이패스 모드가 적용되는 경우, 상기 디코더는 확률 0.5에 기초하여 이진 산술 디코딩을 수행할 수 있다(S660).

[110] 상기 디코더는 디코딩된 빈스트링(bin string)에 대해 역이진화를 수행할 수 있다(S670). 예를 들어, 디코딩된 이진수 형태의 빈(bin)을 입력받아 정수 형태의 신택스 엘리먼트 값으로 변환 출력할 수 있다.

[111]

[112] 최근 비디오 표준에서 고화질, 고프레임율, 고해상도의 영상을 코딩하는 경우, 산술 코딩(arithmetic coding) 알고리즘 등에 내재된 데이터 의존성으로 인해 엔트로피 코딩(entropy coding) 절차가 전체 성능의 병목이 될 가능성이 크다.

많은 비트 데이터가 발생하는 프레임들이 연속되는 경우, 실시간 처리를 위해 여러 프레임들을 버퍼링해야 하므로 상당한 크기의 버퍼 메모리가 필요할 수 있다. 따라서, 엔트로피 코딩의 처리량(throughput) 향상이 요구된다.

[113] 현재 널리 적용되는 엔트로피 코딩(entropy coding) 방법으로 산술 코딩(arithmetic coding) 방법을 들 수 있다. 예를 들어, H.264, HEVC의 경우 이진 심볼(binary symbol)들에 대한 엔트로피 코딩 방법으로 심볼에 대한 확률이 적응적으로 변하는 CABAC(Context-Adaptive Binary Arithmetic Coding)을 적용하고 있다. 여기서, 이진 심볼(binary symbol)은 0 또는 1의 값을 가지는 심볼을 말한다. 반면에, 멀티 심볼(multi-symbol)(즉, 다치 심볼, non-binary symbol) 은 3개 이상의 값을 가질 수 있는 심볼을 말한다.

[114] 산술 코딩 엔진으로 최종 입력되는 가능한 모든 단위 숫자들의 집합은 알파벳(alphabet)으로 지칭될 수 있다. 따라서, 이진 심볼의 경우에는 알파벳이 0과 1로 구성되어 있다고 표현될 수도 있고, 알파벳 심볼로 0과 1이 가능하다고 표현될 수도 있다.

[115] 산술 코딩에서 [0, 1]의 간격(또는 구간)(interval)은 각 심볼 발생의 확률에 비례하는 확률 간격으로 나눠진다. 각 확률 간격의 길이는 해당 심볼(symbol)에 대한 확률 값에 비례하여 결정된다. 간격을 각 심볼들의 확률 간격에 비례하여 나눈 후, 상기 현재 코딩되는 심볼에 대한 간격이 선택되고, 상기 선택된 간격이 다음 심볼을 코딩할 때 사용된다. 심볼들이 코딩되면서 간격의 길이는 계속 작아지기 때문에, 재정규화(renormalization) 절차를 통해 간격 길이를 스케일링(scaling)하여, 간격의 길이가 항상 일정 범위 안에 놓이도록 할 수 있다.

[116] 이하, 본 발명의 설명에 있어 설명의 편의를 위해, CABAC (Context Adaptive Binary Arithmetic Coding) 을 기준으로 예를 들어 설명하나, 본 발명이 이에 한정되는 것은 아니다. 예를 들어, 이진 산술 코딩 대신 비-이진 산술 코딩(non-binary arithmetic coding)이 본 발명에 적용될 수도 있다.

[117] 이진 산술 코딩에서는 확률 간격이 2개로 나눠지고, 멀티 심볼(또는 다치 심볼) 산술 코딩에서는 확률 간격이 사용되는 심볼의 수만큼(즉, 알파벳 심볼의 수만큼) 나눠진다. 나눠지는 간격의 개수를 제외하면 산술 코딩 방법은 이진 산술 코딩과 멀티 심볼 산술 코딩간에 차이가 존재하지 않는다. 다시 말해, 이진 산술 코딩과 멀티 심볼 산술 코딩에 동일한 산술 코딩 방법이 적용될 수 있다.

[118] 또한, 본 발명의 설명에 있어 설명의 편의를 위해, HEVC의 신택스(syntax) 또는 신택스 요소(syntax element)를 기준으로 예를 들어 설명하나, 본 발명이 이에 한정되는 것은 아니다.

[119] 본 발명에서는 엔트로피 코딩의 처리량을 향상시키기 위해 신택스 요소(syntax element)를 기능 블록 단위(function block unit)로 여러 비트 스트림(bit stream)에 매핑하여 코딩하는 방법을 제안한다.

[120] 여기서, 기능 블록은 하나 이상의 신택스 요소가 매핑되어 처리되는(또는 코딩되는) 영역 또는 블록을 나타내며, 상기 기능 블록의 용어는 그 명칭에

한정되는 것은 아니다. 예를 들어, 상기 기능 블록은 서브 블록, 기능 단위, 영역 단위, 기능 영역, 서브 영역, 신택스 요소 그룹 등으로 지칭될 수도 있다.

[121] 또한, 본 발명에서는 기능 블록 단위 또는 신택스 요소 단위로 여러 비트 스트림간 동기화를 수행하는 방법을 제안한다.

[122] 이하 본 발명의 설명에 있어 설명의 편의를 위해, HEVC의 신택스(syntax)를 기준으로 예로 들어 설명하나, 본 발명이 이에 한정되는 것은 아니다.

[123] 도 7은 본 발명이 적용될 수 있는 실시예로서, 레지듀얼 코딩 신택스(residual coding syntax)의 신택스 요소(syntax element)들을 기능 블록 단위(function block unit)로 여러 비트 스트림에 매핑하는 방법을 예시하는 도면이다.

[124] 도 7을 참조하면, 인코더/디코더는 레지듀얼 코딩 루틴(또는 레지듀얼 코딩 신택스)을 복수 개의 서브 루틴(또는 스테이지)으로 분할할 수 있고, 분할된 서브 루틴들을 여러 비트 스트림에 매핑하여 파이프라이닝(Pipelining) 방식으로 코딩할 수 있다. 여기서, 코딩 루틴은 특정 신택스 구조(syntax structure) 상에서 수행되는 일련의 코딩 절차를 나타낸다. 이때, 코딩은 인코딩과 디코딩을 포함한다. 그리고, 서브 루틴은 상기 코딩 루틴을 기능 블록 단위로 분할한 영역을 나타낸다.

[125] 예를 들어, HEVC의 레지듀얼 코딩 루틴을 기능 블록 단위로 분할하면, 7개의 서브 루틴으로 분할될 수 있다. 분할된 7개의 서브 루틴 중 제 1 서브 루틴은 아래의 표 1과 같이 나타낼 수 있다.

[126] [표1]

if(transform_skip_enabled_flag && !cu_transquant_bypass_flag && (log2TrafoSize == 2))	
transform_skip_flag[x0][y0][cIdx]	ae(v)
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)
lastScanPos = 16	
lastSubBlock = (1 << (log2TrafoSize − 2)) * (1 << (log2TrafoSize − 2)) − 1	
do {	
if(lastScanPos == 0) {	
lastScanPos = 16	
lastSubBlock− −	
}	
lastScanPos− −	
xS = ScanOrder[log2TrafoSize − 2][scanIdx][lastSubBlock][0]	
yS = ScanOrder[log2TrafoSize − 2][scanIdx][lastSubBlock][1]	
xC = (xS << 2) + ScanOrder[2][scanIdx][lastScanPos][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][lastScanPos][1]	
} while((xC != LastSignificantCoeffX)		(yC != LastSignificantCoeffY))	

[127] 표 1을 참조하면, 인코더/디코더는 제 1 서브 루틴에서 신택스 요소

transform_skip_flag, last_sig_coeff_x_prefix, last_sig_coeff_y_prefix, last_sig_coeff_x_suffix, last_sig_coeff_y_suffix 를 코딩할 수 있다.

[128] 여기서, transform_skip_flag는 현재 변환 블록에 변환이 적용되는지 여부를 나타내는 신택스 요소를 의미한다. last_sig_coeff_x_prefix는 변환 블록 내에서 스캔 순서상 마지막 의미 있는(significant) 계수의 열 위치의 접두어를 나타낼 수 있다. last_sig_coeff_y_prefix는 변환 블록 내에서 스캔 순서상 마지막 의미 있는 계수의 행 위치의 접두어를 나타낼 수 있다. last_sig_coeff_x_suffix는 변환 블록 내에서 스캔 순서상 마지막 의미 있는 계수의 열 위치 접미사를 나타낼 수 있다. last_sig_coeff_y_suffix는 변환 블록 내에서 스캔 순서상 마지막 의미 있는 계수의 행 위치 접미사를 나타낼 수 있다. 그리고, 상기 의미 있는 계수는 0이 아닌 양자화된 변환 계수를 나타낸다.

[129] 즉, 인코더/디코더는 상기 제 1 서브 루틴에서 현재 변환 블록에 변환이 적용되는지 여부를 나타내는 신택스 요소를 코딩하고, 현재 변환 블록 내 마지막 0이 아닌 변환 계수의 위치를 코딩할 수 있다.

[130] 또한, 분할된 7개의 서브 루틴 중 제 2 서브 루틴은 아래의 표 2와 같이 나타낼 수 있다.

[131] [표2]

xS = ScanOrder[log2TrafoSize − 2][scanIdx][i][0]	
yS = ScanOrder[log2TrafoSize − 2][scanIdx][i][1]	
inferSbDcSigCoeffFlag = 0	
if((i < lastSubBlock) && (i > 0)) {	
coded_sub_block_flag[xS][yS]	ae(v)
inferSbDcSigCoeffFlag = 1	
}	

[132] 표 2를 참조하면, 인코더/디코더는 제 2 서브 루틴에서 신택스 요소 coded_sub_block_flag를 코딩할 수 있다.

[133] 여기서, coded_sub_block_flag는 서브 블록 단위로 0이 아닌 양자화된 변환 계수가 존재하는지 여부를 나타낸다.

[134] 또한, 분할된 7개의 서브 루틴 중 제 3 서브 루틴은 아래의 표 3과 같이 나타낼 수 있다.

[135] [표3]

for(n = (i == lastSubBlock) ? lastScanPos − 1 : 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(coded_sub_block_flag[xS][yS] && (n > 0 || !inferSbDcSigCoeffFlag)) {	
sig_coeff_flag[xC][yC]	ae(v)
if(sig_coeff_flag[xC][yC])	
inferSbDcSigCoeffFlag = 0	
}	

[136] 표 3을 참조하면, 인코더/디코더는 제 3 서브 루틴에서 신택스 요소

sig_coeff_flag를 코딩할 수 있다.

[137] 여기서, sig_coeff_flag는 양자화된 변환 계수가 0보다 큰 값을 갖는지 여부를 나타낼 수 있다.

[138] 또한, 분할된 7개의 서브 루틴 중 제 4 서브 루틴은 아래의 표 4와 같이 나타낼 수 있다.

[139] [표4]

firstSigScanPos = 16	
lastSigScanPos = −1	
numGreater1Flag = 0	
lastGreater1ScanPos = −1	
for(n = 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0] yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC]) {	
if(numGreater1Flag < 8) {	
coeff_abs_level_greater1_flag[n]	ae(v)
numGreater1Flag++	
if(coeff_abs_level_greater1_flag[n] && lastGreater1ScanPos == −1)	
lastGreater1ScanPos = n	
}	
if(lastSigScanPos == −1)	
lastSigScanPos = n	
firstSigScanPos = n	
}	
}	

[140] 표 4를 참조하면, 인코더/디코더는 제 4 서브 루틴에서 신택스 요소 coeff_abs_level_greater1_flag 를 코딩할 수 있다.

[141] 여기서, coeff_abs_level_greater1_flag 는 양자화된 변환 계수의 절대값이 1보다 큰 값을 가지는지 여부를 나타낼 수 있다.

[142] 또한, 분할된 7개의 서브 루틴 중 제 5 서브 루틴은 아래의 표 5와 같이 나타낼 수 있다.

[143] [표5]

signHidden = (lastSigScanPos − firstSigScanPos > 3 && !cu_transquant_bypass_flag)	
if(lastGreater1ScanPos != −1)	
coeff_abs_level_greater2_flag[lastGreater1ScanPos]	ae(v)

[144] 표 5를 참조하면, 인코더/디코더는 제 5 서브 루틴에서 신택스 요소 coeff_abs_level_greater2_flag 를 코딩할 수 있다.

[145] 여기서, coeff_abs_level_greater2_flag 는 양자화된 변환 계수의 절대값이 2보다 큰 값을 가지는지 여부를 나타낼 수 있다.

[146] 또한, 분할된 7개의 서브 루틴 중 제 6 서브 루틴은 아래의 표 6과 같이 나타낼 수 있다.

[147] [표6]

for(n = 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC] && (!sign_data_hiding_enabled_flag || !signHidden || (n != firstSigScanPos)))	
coeff_sign_flag[n]	ae(v)
}	

[148] 표 6을 참조하면, 인코더/디코더는 제 6 서브 루틴에서 신택스 요소 coeff_sign_flag 를 코딩할 수 있다.

[149] 여기서, coeff_sign_flag 는 양자화된 변환 계수의 부호를 나타낼 수 있다.

[150] 또한, 분할된 7개의 서브 루틴 중 제 7 서브 루틴은 아래의 표 7과 같이 나타낼 수 있다.

[151] [표7]

numSigCoeff = 0	
sumAbsLevel = 0	
for(n = 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC]) {	
baseLevel = 1 + coeff_abs_level_greater1_flag[n] + coeff_abs_level_greater2_flag[n]	
if(baseLevel == ((numSigCoeff < 8) ? ((n == lastGreater1ScanPos) ? 3 : 2) : 1))	
coeff_abs_level_remaining[n]	ae(v)
TransCoeffLevel[x0][y0][cIdx][xC][yC] = (coeff_abs_level_remaining[n] + baseLevel) * (1 − 2 * coeff_sign_flag[n])	
if(sign_data_hiding_enabled_flag && signHidden) {	
sumAbsLevel += (coeff_abs_level_remaining[n] + baseLevel)	
if((n == firstSigScanPos) && ((sumAbsLevel % 2) == 1))	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = −TransCoeffLevel[x0][y0][cIdx][xC][yC]	
}	
numSigCoeff++	
}	
}	

[152] 표 7을 참조하면, 인코더/디코더는 제 7 서브 루틴에서 신택스 요소 coeff_abs_level_remaining 를 코딩할 수 있다.

[153] 여기서, coeff_abs_level_remaining 는 양자화된 변환 계수의 값의 나머지 값의 절대값을 나타낼 수 있다.

[154] 인코더/디코더는 제 1 서브 루틴에서 변환 유닛(TU: transform unit) 단위로 상기 TU에 변환이 적용되는지 여부 및 마지막 0이 아닌 변환 계수의 위치를 코딩(또는 파싱)할 수 있고, 인코더/디코더는 이후의 서브 루틴(즉 제 2 내지 제 7 서브 루틴)에서 계수 그룹(CG: cefficient group)(또는 서브 그룹) 단위로 표 2 내지 표

6에서 설명한 신택스 요소들을 코딩할 수 있다.

[155] 도 7을 참조하면, 인코더/디코더는 TU 단위로 수행되는 제 1 서브 루틴을 제 1 비트 스트림(이하, BS1)(701)에 매핑하고, 제 2 서브 루틴부터 파이프라이닝 방식으로 CG 단위로 비트 스트림을 번갈아 가며 코딩할 수 있다. 다시 말해, 인코더/디코더는 제 1 서브 루틴 이후 CG 단위로 수행되는 각 서브 루틴들을 여러 비트 스트림에서 코딩할 수 있다.

[156] 구체적으로, 인코더/디코더는 첫 번째 CG의 제 2 서브 루틴을 BS1(701)에서 수행한 후에 다음 CG의 제 2 서브 루틴을 제 2 비트 스트림(이하, BS2)(702)에서 수행할 수 있다. 인코더/디코더는 두 번째 CG의 제 2 서브 루틴을 BS2(702)에서 수행한 후에 다음 CG의 제 2 서브 루틴을 제 3 비트 스트림(BS3)(703)에서 수행할 수 있다.

[157] 위와 같은 방식으로 인코더/디코더는 제 2 서브 루틴부터 제 7 서브 루틴까지 CG 단위로 비트 스트림을 번갈아가며 서브 루틴의 신택스 요소를 코딩할 수 있다.

[158] 이 경우, 인코더/디코더는 이전에 코딩된 정보에 대한 데이터 의존성(data dependency)을 고려하여 서브 루틴이 전환되는 시점(즉, 도 7에서 점선으로 도시된 시점)에 비트 스트림들간 동기화를 수행할 수 있다. 다시 말해, 인코더/디코더는 서브 루틴(또는 기능 블록 단위)을 주기로 여러 비트 스트림간 동기화를 수행할 수 있다.

[159] 여기서, 데이터 의존성이 존재하는 경우는, 예를 들어, 동일한 컨텍스트 모델을 사용하는 경우, 이전에 코딩된 정보를 참조하는 경우 등이 있을 수 있다. 상기 서브 루틴이 전환되는 시점에서 비트 스트림들간 동기화를 수행함으로써, 인코더/디코더는 코딩 순서를 지키면서 각각의 서브 루틴을 병렬적으로 코딩(또는 수행)할 수 있다.

[160] 반면에, 인코더/디코더는 코딩 순서를 지키면서 병렬적으로 코딩하기 위해 신택스 요소 단위로 동기화를 수행할 수도 있다. 아래의 도면을 참조하여 설명한다.

[161] 도 8은 본 발명이 적용될 수 있는 실시예로서, 신택스 요소(syntax element) 단위로 여러 비트 스트림간 동기화를 수행하는 방법을 예시하는 레지듀얼 코딩(residual coding) 신택스이다.

[162] 전술한 바와 같이, 인코더/디코더는 레지듀얼 코딩 루틴을 복수 개의 서브 루틴으로 분할할 수 있고, 분할된 서브 루틴들을 여러 비트 스트림에 매핑하여 파이프라이닝 방식으로 코딩할 수 있다.

[163] 본 발명의 일 실시예에서, 인코더/디코더는 데이터 의존성을 고려하여 신택스 요소 단위로 여러 비트 스트림간 동기화를 수행할 수 있다.

[164] 구체적으로, 인코더/디코더는 여러 비트 스트림간 동기화를 수행하기 위해 현재 신택스 요소 또는 현재 처리 영역(또는 현재 처리 블록)에 대한 락(lock)을 획득하고 반환하는 절차를 수행할 수 있다.

[165] 도 8(a)를 참조하면, 인코더/디코더는 각 CG(또는 기능 블록) 에서 신택스 요소 coded_sub_block_flag를 배타적으로 코딩하기 위해 락을 획득하고 해제할 수 있다(S801).

[166] 여기서, '(LA)'는 락 획득(lock acquire)으로서 해당 신택스 요소 또는 처리 영역의 코딩을 배타적으로(exclusive)하게 수행하기 위해 락을 획득하는 동작을 나타낸다. 현재 비트 스트림에서 현재 CG에 대한 락을 획득하는 경우, 다른 비트 스트림(또는 CG 파이프라인, CG 스레드, CG CABAC)에서는 해당 신택스 요소 또는 처리 영역의 코딩을 시작하지 않고 코딩을 멈출 수 있다.

[167] 그리고, '(LR)'는 락 해제(lock release)로서 획득했던 락을 해제하여 다른 비트 스트림에서 락을 획득할 수 있도록 하는 동작을 나타낸다.

[168] 이때, 인코더/디코더는 CG들간 처리 순서(또는 코딩 순서)를 지키면서 락을 획득하고 반환하는 동작을 수행할 수 있다. 예를 들어, i번째 CG에서 신택스 요소 또는 영역의 처리(또는 코딩)가 i-1번째 CG의 신택스 요소 또는 영역의 처리가 끝나야만 가능한 경우, 인코더/디코더는 어떤 CG가 락을 해제했는지를 구분하여 다음 CG에서 락을 획득하도록 할 수 있다.

[169] 만약 CG 단위로 수행되는 서브 루틴들이 여러 bit-stream들에 번갈아 가면서 코딩되는 경우, 인코더/디코더는 이전 비트 스트림(또는 CG)에서 락을 획득하고 반환했는지를 체크하여 현재 비트 스트림에 대한 락의 획득을 시도할 수 있다.

[170] 도 8(b)를 참조하면, 인코더/디코더는 CG 마다 신택스 요소 sig_coeff_flag를 코딩하는 CG loop(또는 서브 루틴)에 앞서 락을 획득할 수 있다(S802).

[171] 즉, 인코더/디코더는 S801 단계에서와 같이, 신택스 요소를 코딩하기에 전에 락을 획득하고 상기 신택스 요소를 코딩한 후 락을 해제하여 여러 비트 스트림간 동기화를 수행할 수도 있고, S802 단계에서와 같이, CG 마다 수행되는 CG loop를 수행하기 전에 락을 획득하고 상기 CG loop가 종료한 후 락을 해제하여 여러 비트 스트림간 동기화를 수행할 수도 있다.

[172] 신택스 요소 마다 락을 획득/해제하는 경우, 각 CG의 신택스 요소간 존재할 수 있는 데이터 의존성을 효과적으로 고려하면서 병렬적으로 코딩할 수 있다. 다시 말해, 신택스 요소 마다 동기화를 수행함으로써, 인코더/디코더는 각 CG의 신택스 요소간의 순서를 지키면서 코딩할 수 있다.

[173] 반면에, CG loop 마다 락을 획득/해제하는 경우, 해당 CG loop를 수행하기 전에 락을 획득함으로써 여러 비트 스트림간 동기화를 수행할 수 있고, CG loop를 병렬적으로 수행할 수 있다. 이때, CG loop 내 각 CG에 대한 코딩은 파이프 라이닝 방식으로 여러 비트 스트림에서 수행될 수 있다.

[174] 도 8(c)를 참조하면, 인코더/디코더는 동기화를 수행함 없이, 즉, 락의 획득 및 반환 절차를 수행하지 않고, 신택스 요소 coeff_sign_flag와 coeff_abs_level_remaining를 코딩할 수 있다(S803, S804).

[175] 예를 들어, 신택스 요소 coeff_sign_flag와 coeff_abs_level_remaining에 바이패스 코딩이 적용되는 경우, 데이터 의존성이 존재하지 않을 수 있다. 이러한 경우,

인코더/디코더는 상기 coeff_sign_flag와 coeff_abs_level_remaining를 코딩할 때, 동기화를 적용하지 않음으로써, CG 단위(또는 기능 블록 단위)로 수행되는 서브 루틴의 병렬 수행성(parallelism)을 향상시킬 수 있다.

[176] 도 9는 본 발명이 적용되는 일 실시예로서, 레지듀얼 코딩(residual coding)의 신택스 요소(syntax element)를 기능 블록 단위(function block unit)로 여러 비트 스트림에서 코딩하는 방법을 예시하는 신택스이다.

[177] 본 발명의 일 실시예에서, 인코더/디코더는 각 기능 블록(또는 서브 루틴)의 신택스 요소를 코딩한 후, 비트 스트림 전환하여 복수 개의 변환 유닛을 여러 비트 스트림에서 동시에 처리할 수 있다.

[178] HEVC의 경우를 예로 들면, TU의 크기에 따라 TU 내 CG의 개수는 가변적이다. 앞서 도 7 및 도 8에서 설명한 방법은, CG의 개수가 하나 이상인 경우 엔트로피 코딩의 처리량(throughput)이 향상될 수 있고, 비트 스트림의 개수보다 CG의 개수가 적지 않은 경우 엔트로피 코딩의 처리량이 극대화 될 수 있다.

[179] 반면, TU의 크기에 따라 병렬화(또는 파이프라이닝)를 적용할 수 있는 CG의 개수가 적은 경우에는, CG 단위(또는 서브 루틴 단위)로 비트 스트림을 전환함으로써, 여러 TU 또는 여러 TU들에 속한 CG들을 병렬적으로 처리할 수 있고 이로 인해 엔트로피 코딩의 처리량이 향상될 수 있다.

[180] 구체적으로, 인코더/디코더는 현재의 TU 내에서 CG 단위로 수행되는 서브 루틴에 대해 비트 스트림을 할당한 후, 비트 스트림을 전환하여 그 다음 TU 내에서 CG 단위로 수행되는 서브 루틴 이전(앞서 도 7에서 제 1 서브 루틴)까지 연이어 비트 스트림을 할당할 수 있다. 즉, 인코더/디코더는 현재 TU의 CG 루프(앞서 도 7에서 제 2 내지 제 7 서브 루틴)와 다음 TU의 CG 루프 전(앞서 도 7에서 제 1 서브 루틴)까지의 코딩을 병렬화할 수 있고, 다음 TU의 GG 루프에 대해 연속적으로 비트 스트림을 할당하여 여러 비트스트림들에 대한 파이프라인들을 쉬지 않고 수행시킬 수 있으며, 파이프라이닝 스톨이 발생하는 구간을 최소화할 수 있다. 여기서, 파이프라이닝 스톨은 파이프 라인에서 발생하는 코딩 지연을 의미한다.

[181]

[182] 앞서 도 7에서 전술한 바와 같이, TU 내에서 제 1 서브 투틴과 CG 루프(제 2 내지 제 7 서브 루틴)은 동시에 수행될 수 없다. 본 실시예에 따르면, 인코더/디코더는 현재 TU에 대한 CG 루프를 수행하면서, 동시에 다음 TU에 대한 제 1 서브 루틴을 수행할 수 있기 때문에, 상기 다음 TU의 CG 루프를 시작하기까지 발생할 수 있는 유휴 시간을 줄일 수 있다.

[183] 도 9를 참조하면, 인코더/디코더는 CG 루프를 통해 CG 단위로 신택스 요소를 코딩한 후 다음 비트 스트림으로 전환할 수 있다(S901).

[184] 예를 들어, 인코더/디코더는 비트 스트림의 전환을 나타내는 set_next_bitstream()을 호출함으로써 비트 스트림을 전환할 수 있다. 상기 set_next_bitstream() 구문이 신택스 상에 표현됨으로써, 인코더/디코더는 여러

비트 스트림을 이용하여 병렬적으로 엔트로피 코딩을 수행할 수 있다.

[185] 이 경우 CG 루프인 for 루프의 반복(iteration) 마다 비트 스트림을 전환되기 때문에, 마지막 CG에 대한 서브 루틴 후에도 비트 스트림이 전환될 수 있다. 따라서, 다음 TU의 CG 루프 이전까지의 서브 루틴과 다음 CG 루프의 첫 번째 반복(iteration)에 해당하는 서브 루틴을 상기 전환된 비트 스트림에서 코딩할 수 있다.

[186] 아래의 표 8의 TU의 신택스를 참조하여 설명한다.

[187] [표8]

transform_unit(x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx) {	**Descriptor**
if(cbf_luma[x0][y0][trafoDepth] || cbf_cb[x0][y0][trafoDepth] || cbf_cr[x0][y0][trafoDepth]) {	
if(cu_qp_delta_enabled_flag && !IsCuQpDeltaCoded) {	
cu_qp_delta_abs	ae(v)
if(cu_qp_delta_abs)	
cu_qp_delta_sign_flag	ae(v)
}	
if(cbf_luma[x0][y0][trafoDepth])	
residual_coding(x0, y0, log2TrafoSize, 0)	
if(log2TrafoSize > 2) {	
if(cbf_cb[x0][y0][trafoDepth])	
residual_coding(x0, y0, log2TrafoSize − 1, 1)	
if(cbf_cr[x0][y0][trafoDepth])	
residual_coding(x0, y0, log2TrafoSize − 1, 2)	
} else if(blkIdx == 3) {	
if(cbf_cb[xBase][yBase][trafoDepth])	
residual_coding(xBase, yBase, log2TrafoSize, 1)	
if(cbf_cr[xBase][yBase][trafoDepth])	
residual_coding(xBase, yBase, log2TrafoSize, 2)	
}	
}	
}	

[188] 표 8을 참조하면, 예를 들어, TU에서 여러 개의 레지듀얼 코딩 루틴(또는 레지듀얼 코딩 신택스)이 호출되고, 각각의 레지듀얼 코딩 루틴 내에서 하나의 CG만 코딩되는 경우를 가정하여 설명한다.

[189] 이 경우, 본 실시예에서 제안하는 방법에 따르면, 인코더/디코더는 다음 TU의 CG 루프 전까지의 코딩(즉, 신택스 요소 transform_skip_flag, last_sig_coeff_x_prefix, last_sig_coeff_y_prefix, last_sig_coeff_x_suffix, last_sig_coeff_y_suffix의 코딩)을 현재 TU의 CG 루프의 코딩과 동시에 수행할 수 있으며, 상기 다음 TU의 CG 루프에 대한 코딩도 해당 비트 스트림에서 연이어 수행할 수 있다.

[190] 이상에서, 레지듀얼 코딩의 신택스 요소를 여러 비트 스트림에 매핑하여 병렬적으로 코딩하는 방법을 설명하였다. 전술한 기능 블록 단위 병렬 코딩 방법은 레지듀얼 코딩 루틴 이외의 코딩 루틴(또는 파싱 루틴)들에 대해서도

동일한 방식으로 적용될 수 있다. 아래의 도면을 참조하여 설명한다.

[191] 도 10은 본 발명이 적용될 수 있는 실시예로서, 코딩 유닛(coding unit)의 신택스 요소를 여러 비트 스트림에서 코딩하는 방법을 예시하는 신택스이다.

[192] 본 발명의 일 실시예에서, 인코더/디코더는 코딩 유닛(CU: coding_unit) 루틴(또는 CU 신택스)이 수행되는 스레드(thread)를 여러 비트스트림에 매핑하여 병렬적으로 코딩할 수 있다. 여기서, 스레드는 엔트로피 인코더/디코더가 독립적으로 처리하는 작업 단위를 말하며, 상기 스레드는 그 용어의 명칭에 한정되는 것은 아니다. 또한, 인코더/디코더는 라운드 로빈(round-robin) 방식으로 기능 블록 또는 스레드들을 여러 비트 스트림에 할당할 수 있다.

[193] 먼저, 인코더/디코더는 현재 CU가 하위 CU(또는 하위 심도의 CU)으로 분할되는지 여부를 나타내는 신택스 요소 split_cu_flag를 코딩한다(S1001).

[194] 현재 CU이 하위 CU로 분할되지 않는 경우, CU 단위로 수행되는 코딩 루틴을 수행하기 위한 CU 스레드(thread)를 스레드 큐(thread queue)에 등록한다(S1002).

[195] 즉, 인코더/디코더는 CU 루틴을 병렬적으로 수행하기 위하여 CTU 신택스에서 CU 신택스를 직접 호출하지 않고, CU 루틴을 하나의 스레드로서 스레드 큐에 등록할 수 있다. 이후, 인코더/디코더는 상기 스레드 큐에 저장된(또는 상기 스레드 큐에 등록된) 복수 개의 CU 스레드들을 각각 여러 비트 스트림에 매핑하여 병렬적으로 코딩할 수 있다.

[196] 도 11은 본 발명이 적용될 수 있는 실시예로서, 예측 유닛(prediction unit)의 신택스 요소를 여러 비트 스트림에서 코딩하는 방법을 예시하는 신택스이다.

[197] 본 발명의 일 실시예에서, 인코더/디코더는 예측 유닛(PU: prediction unit) 루틴이 수행되는 스레드(thread)를 여러 비트스트림에 매핑하여 병렬적으로 코딩할 수 있다.

[198] 인코더/디코더는 현재 CU에 CU 스킵(skip)이 적용되는 경우, PU 단위로 수행되는 코딩 루틴을 수행하기 위한 PU 스레드(thread)를 스레트 큐에 등록한다(S1101).

[199] 여기서, CU 스킵이란 CU 신택스에서 머지(merge)를 위한 인덱스 정보 외에는 추가적인 신택스 요소가 시그널링되지 않는 것을 의미한다. 이때, 현재 CU에 CU 스킵이 적용되는지 여부는 cu_skip_flag를 통해 인코더로부터 시그널링될 수 있다. cu_skip_flag의 값이 1 이면 현재 CU에 스킵 모드가 적용되는 것을, cu_skip_flag의 값이 0이면 스킵 모드가 적용되지 않는 것을 의미할 수 있다. 디코더는 cu_skip_flag를 파싱하여 현재 CU에 CU 스킵이 적용되는지 여부를 판단할 수 있다.

[200] 현재 CU의 예측 모드가 인터 모드(inter mode)이고 현재 CU의 분할 모드(PartMode)가 PART_2Nx2N 인 경우, 인코더/디코더는 신택스 요소 merge_flag를 코딩할 수 있고(S1102), PU의 PU 스레드를 스레드 큐에 등록할 수 있다(S1103).

[201] 만약, 현재 CU의 분할 모드(PartMode)가 PART_2Nx2N이 아닌 경우,

인코더/디코더는 현재 CU의 분할 모드에 따라 분할된 PU의 PU 스레드들을 스레드 큐에 등록할 수 있다.

[202] 즉, 인코더/디코더는 PU 루틴을 병렬적으로 수행하기 위하여 CU 신택스에서 PU 신택스(또는 PU 스레드, PU 신택스 루틴, PU 루틴)를 직접 호출하지 않고, PU 루틴을 하나의 스레드로서 스레드 큐에 등록할 수 있다. 이후, 인코더/디코더는 상기 스레드 큐로부터 출력되는(또는 스레드 큐에 등록된) 복수 개의 PU 스레드들을 각각 여러 비트 스트림에 매핑하여 코딩할 수 있다.

[203] 또한, S1102 단계에서와 같이 인코더/디코더는 현재 CU의 분할 모드가 PART_2Nx2N인 경우, 신택스 merge_flag를 CU 신택스 내에서 코딩함으로써, PU 스레드들을 독립적으로 코딩할 수 있다.

[204] 앞서 도 11에서는 CU 신택스에서 PU 스레드를 생성하여 스레드 큐에 등록하고, 여러 스레드들을 상기 스레드 큐로부터 꺼내 여러 비트 스트림에서 병렬적으로 코딩하는 방법을 설명하였다. 이하에서 아래의 도면을 참조하여 PU 신택스를 호출하면서 비트 스트림을 전환하여 PU 스레드(또는 PU 루틴)을 병렬적으로 코딩하는 방법을 설명한다.

[205] 도 12는 본 발명이 적용될 수 있는 실시예로서, 예측 유닛(prediction unit)의 신택스 요소를 여러 비트 스트림에서 코딩하는 방법을 예시하는 신택스이다.

[206] 본 발명의 일 실시예에서, 인코더/디코더는 CU 신택스에서 PU 신택스를 호출하고 비트 스트림을 전환하여 PU 신택스(또는 PU 스레드, PU 루틴, PU 신택스 루틴)들을 여러 비트스트림에 매핑하여 병렬적으로 코딩할 수 있다.

[207] 도 12(a)를 참조하면, 인코더/디코더는 현재 CU에 CU 스킵이 적용되는 경우, PU 신택스를 호출하고, 비트 스트림을 전환할 수 있다(S1201).

[208] 예를 들어, 인코더/디코더는 비트 스트림의 전환을 나타내는 set_next_bitstream()을 호출함으로써 비트 스트림을 전환할 수 있다. 상기 set_next_bitstream() 구문이 신택스 상에 표현됨으로써, 인코더/디코더는 여러 비트 스트림을 이용하여 병렬적으로 엔트로피 코딩을 수행할 수 있다.

[209] 즉, 인코더/디코더는 CU 신택스에서 PU 신택스를 호출할 때마다 비트 스트림을 변경하여 PU(또는 PU 스레드)의 코딩과 상기 CU 신택스 내 나머지 신택스 요소의 코딩을 병렬화할 수 있다. 이때, 라운드 로빈(round-robin) 방식으로 여러 비트 스트림 중 다음 비트 스트림이 결정될 수 있다.

[210] PU의 분할 모드가 PART_2NxN인 경우, 인코더/디코더는 PU 신택스를 호출하고, 비트 스트림을 전환할 수 있다(S1202).

[211] 만약, PU의 분할 모드가 PART_2Nx2N이 아닌 경우, PU 신택스는 CU 신택스 내에서 두 번 이상 호출될 수 있고, 두 개 이상의 PU 스레드들이 생성되어 여러 비트 스트림에서 병렬적으로 수행될 수 있다.

[212] 도 13은 본 발명이 적용될 수 있는 실시예로서, 코딩 유닛(coding unit)의 신택스 요소를 여러 비트 스트림에서 코딩하는 방법을 예시하는 도면이다.

[213] 도 13를 참조하면, CU 루틴은 3개의 스테이지(또는 서브 루틴)(1301, 1302,

1303)으로 분할될 수 있다. 인코더/디코더는 상기 분할된 3개의 스테이지를 파이프라이닝 방식으로 코딩할 수 있다.

[214] 구체적으로, 제 1 스테이지(1301)에서 인코더/디코더는 신택스 요소 cu_transquant_bypass_flag와 cu_skip_flag를 코딩할 수 있다. 만약, cu_transquant_bypass_flag 값이 1이면 스케일링 및 변환 절차(scaling and transform process)와 인-루프 필터 절차(in-loop filter process)를 건너뛸 수 있다. 그리고, cu_skip_flag는 현재 CU에 스킵 모드가 적용되는지 여부를 나타낼 수 있다.

[215] 제 2 스테이지(1302)에서 신택스 요소 cu_skip_flag 값이 1인 경우, 인코더/디코더는 PU 스레드(prediction unit thread)를 스레트 큐에 등록할 수 있다.

[216] 제 2 스테이지(1302)에서 신택스 요소 cu_skip_flag 값이 0인 경우, 인코더/디코더는 신택스 요소 pred_mode_flag와 part_mode를 코딩할 수 있다. 여기서, pred_mode_flag는 현재 CU에 적용된 예측 모드를 나타낼 수 있다. 즉, 상기 pred_mode_flag 값이 0인 경우에는 인터 예측 모드로 코딩됨을, 상기 pred_mode_flag 값이 1인 경우에는 인트라 예측 모드로 코딩됨을 나타낼 수 있다. part_mode는 현재 CU의 분할 모드를 나타낼 수 있다.

[217] 만약, 현재 블록의 예측 모드가 인트라 예측 모드인 경우, 인코더/디코더는 신택스 요소 pcm_flag, pcm_alignment_zero_bit, prev_intra_luma_pred_flag, mpm_idx, rem_intra_luma_pred_mode, intra_chroma_pred_mode를 코딩할 수 있다.

[218] 여기서, pcm_flag값이 1인 경우 휘도 성분의 CU이 pcm_sample 신택스(syntax)가 존재하고, transform_tree신택스는 존재하지 않음을 나타낸다. 인코더/디코더는 비트 스트림에서 현재 위치가 한 바이트(byte)의 경계에 있지 않은 경우 pcm_alignment_zero_bit를 코딩하며, 이때, pcm_alignment_zero_bit의 값은 0이다. prev_intra_luma_pred_flag 값이 1인 경우 현재 PU의 인트라 예측 모드가 MPM(Most Probable Mode)모드에 포함되는 것을 나타내며, 0인 경우는 현재 PU의 인트라 예측 모드가 MPM(Most Probable Mode) 모드에 포함되지 않는 것을 나타낸다. MPM(Most Probable Mode)모드에 포함되지 않은 나머지 예측 모드의 경우, 인코더/디코더는 rem_intra_luma_pred_mode를 코딩할 수 있다. 그리고, intra_chroma_pred_mode는 PU 단위로 색차 성분의 예측 모드를 나타낼 수 있다.

[219] 만약, 현재 블록의 예측 모드가 인트라 예측 모드가 아닌 경우(또는 인터 예측 모드인 경우), 인코더/디코더는 신택스 요소 merge_flag를 코딩할 수 있고, PU 스레드(thread)를 스레트 큐에 등록할 수 있다.

[220] 제 2 스테이지(1302)에서만 PU 스레드를 스레드 큐에 등록함으로써, PU들간 스레드 큐에 등록되는 순서(또는 이슈(issue)되는 순서)를 지킬 수 있다.

[221] 제 3 스테이지(1303)에서 현재 CU의 예측 모드가 인트라 예측 모드가 아니고, 현재 CU이 머지 모드임과 동시에 분할 모드가 PART_2Nx2N인 경우가 아니라면, 인코더/디코더는 신택스 요소 rqt_root_cbf를 코딩할 수 있다. 여기서, rqt_root_cbf 현재 CU을 위한 변환 트리 신택스(transform_tree syntax)의 존재 여부를 나타낼

수 있다.

[222] 그리고, 제 3 스테이지(1303)에서 인코더/디코더는 변환 트리 스레드(transform tree thread)를 스레드 큐(thread queue)에 등록할 수 있다.

[223] 도 14는 본 발명이 적용될 수 있는 실시예로서, 신택스 요소(syntax element) 단위로 여러 비트 스트림간 동기화를 수행하는 방법을 예시하는 코딩 유닛(coding unit) 신택스이다.

[224] 본 발명의 일 실시예에서, 인코더/디코더는 데이터 의존성을 고려하여 신택스 요소 단위로 여러 비트 스트림간 동기화를 수행할 수 있다. 구체적으로, 인코더/디코더는 여러 비트 스트림간 동기화를 수행하기 위해 현재 신택스 요소 또는 현재 처리 영역(또는 현재 처리 블록)에 대한 락(lock)을 획득하고 반환하는 절차를 수행할 수 있다.

[225] 도 14(a)를 참조하면, 인코더/디코더는 각 CU(또는 CU 스레드)에서 신택스 요소 cu_transquant_bypass_flag를 배타적으로 코딩하기 위해 락을 획득하고 해제할 수 있다(S1401).

[226] 즉, 인코더/디코더는 앞서 도 8에서 설명한 바와 같이 현재 비트 스트림에서 현재 CU에 대한 락을 획득하는 경우, 다른 비트 스트림(또는 다른 CU 스레드)에서 해당 신택스 요소 또는 처리 영역의 코딩을 시작하지 않고 코딩을 멈출 수 있다.

[227] 이때, 인코더/디코더는 CU들간 처리 순서(또는 코딩 순서)를 지키면서 락을 획득하고 반환하는 동작을 수행할 수 있다.

[228] S1401 단계와 동일하게 인코더/디코더는 CU의 신택스 요소들을 코딩하는 과정에서 신택스 요소 단위로 락을 획득/해제하여 여러 비트 스트림간 동기화를 수행할 수 있다.

[229] 전술한 바와 같이, 인코더/디코더는 현재 CU의 분할 모드가 PART_2Nx2N인 경우, PU 스레드들을 독립적으로 코딩하기 위해 merge_flag를 CU 신택스 내에서 코딩할 수 있다(S1402).

[230] 인코더/디코더는 PU 스레드가 생성(또는 이슈)되는 순서를 지키기 위해 PU의 분할 모드에 따른 PU 스레드 생성에 앞서 여러 비트 스트림간 동기화를 수행할 수 있다(S1403). 다시 말해, 인코더/디코더는 분할 모드에 따른 PU 스레드를 생성하기에 앞서 락을 획득할 수 있다.

[231] 도 15는 본 발명이 적용될 수 있는 실시예로서, 예측 유닛(prediction unit)의 신택스 요소를 여러 비트 스트림에서 코딩하는 방법을 예시하는 신택스이다.

[232] 도 15를 참조하면, cu_skip_flag의 값이 0이고 현재 PU의 분할 모드가 PART_2Nx2N인 경우, CU 신택스에서 신택스 요소 merge_flag가 이미 코딩되었을 수 있다(앞서 도 11의 S1102, 도 14의 S1402 참조).

[233] 따라서, 인코더/디코더는 cu_skip_flag의 값이 0이고 현재 PU의 분할 모드가 PART_2Nx2N이 아닌 경우, merge_flag를 코딩할 수 있다(S1501).

[234] 도 16은 본 발명이 적용될 수 있는 실시예로서, PU(prediction unit)의 신택스

요소를 여러 비트 스트림에서 코딩하는 방법을 예시하는 도면이다.

[235] 도 16을 참조하면, PU 루틴은 3개의 스테이지(또는 서브 루틴)(1601, 1602, 1603)으로 분할될 수 있다. 인코더/디코더는 상기 분할된 3개의 스테이지를 파이프라이닝 방식으로 여러 비트 스트림에 코딩할 수 있다.

[236] 구체적으로, 제 1 스테이지(1601)에서 인코더/디코더는 cu_skip_flag의 값이 1일 경우, 신택스 요소 merge_idx를 코딩할 수 있다. 만약, cu_skip_flag의 값이 0일 경우, 인코더/디코더는 신택스 요소 merge_flag를 코딩할 수 있다.

[237] 그리고, 제 1 스테이지(1601)에서 인코더/디코더는 merge_flag 값이 1일 경우, merge_idx를 코딩할 수 있다. 만약, merge_flag 값이 1일 경우, 인코더/디코더는 신택스 요소 inter_pred_idc를 코딩할 수 있다. 여기서, inter_pred_idc는 현재 PU에 리스트0(list0), 리스트1(list1) 또는 양방향 예측이 사용되는지 여부를 나타낸다.

[238] 제 2 스테이지(1602)에서 인코더/디코더는 inter_pred_idc 값이 리스트1 방향이 아닌 경우, 신택스 요소 ref_idx_l0, mvd_coding, mvp_l0_flag를 코딩할 수 있다. 여기서, ref_idx_l0는 리스트0 방향의 참조 픽쳐 인덱스를 나타내며, mvp_l0_flag는 리스트 0 방향의 움직임 벡터 예측값 인덱스를 나타낸다. mvd_coding는 신택스 요소 단위로 동기화를 수행하는 경우, PU 신택스 상에서 코딩될 수 있다. 이하, 자세히 후술한다.

[239] 제 2 스테이지(1602)에서 인코더/디코더는 inter_pred_idc 값이 리스트1 방향인 경우, 신택스 요소 ref_idx_l1, mvd_coding, mvp_l1_flag를 코딩할 수 있다. 여기서, ref_idx_l1는 리스트1 방향의 참조 픽쳐 인덱스를 나타내며, mvp_l1_flag는 리스트 1 방향의 움직임 벡터 예측값 인덱스를 나타낸다.

[240] 제 3 스테이지(1603)에서 인코더/디코더는 inter_pred_idc 값이 양방향 예측인 경우, 신택스 요소 ref_idx_l1, mvd_coding, mvp_l1_flag를 코딩할 수 있다.

[241] 이때, ref_idx_l0와 ref_idx_l1간, mvp_l0_flag와 mvp_l1_flag간 데이터 의존성이 존재하지 않으므로, 인코더/디코더는 상기 신택스 요소들을 동시에 코딩할 수 있다.

[242] 도 17은 본 발명이 적용될 수 있는 실시예로서, 신택스 요소(syntax element) 단위로 여러 비트 스트림간 동기화를 수행하는 방법을 예시하는 예측 유닛(coding unit) 신택스이다.

[243] 본 발명의 일 실시예에서, 인코더/디코더는 데이터 의존성을 고려하여 신택스 요소 단위로 여러 비트 스트림간 동기화를 수행할 수 있다.

[244] 구체적으로, 인코더/디코더는 여러 비트 스트림간 동기화를 수행하기 위해 현재 신택스 요소 또는 현재 처리 영역(또는 현재 처리 블록)에 대한 락(lock)을 획득하고 반환하는 절차를 수행할 수 있다.

[245] 앞서 도 8에서 설명한 바와 같이, '(LA)'는 락 획득(lock acquire)으로서 해당 신택스 요소 또는 처리 영역의 코딩을 배타적으로(exclusive)하게 수행하기 위해 락을 획득하는 동작을 나타낸다. 현재 비트 스트림에서 현재 CG에 대한 락을 획득하는 경우, 다른 비트 스트림(또는 PU 파이프라인, PU 스레드, PU

CABAC)에서는 해당 신택스 요소 또는 처리 영역의 코딩을 시작하지 않고 코딩을 멈출 수 있다.

[246] 그리고, '(LR)'는 락 해제(lock release)로서 획득했던 락을 해제하여 다른 비트 스트림에서 락을 획득할 수 있도록 하는 동작을 나타낸다.

[247] 이때, 인코더/디코더는 PU들간 처리 순서(또는 코딩 순서)를 지키면서 락을 획득하고 반환하는 동작을 수행할 수 있다.

[248] 또한, 인코더/디코더는 신택스 요소 단위의 동기화를 효과적으로 수행하기 위하여 PU 신택스(또는 PU 루틴) 내에서 움직임 벡터 차분값 코딩(mvd_coding)을 수행할 수 있다.

[249] 도 18은 본 발명이 적용될 수 있는 실시예로서, 코딩 트리 유닛(CTU)으로부터 계층적으로 분할되는 유닛들을 여러 비트 스트림에 매핑하여 코딩하는 방법을 설명하기 위한 도면이다.

[250] 도 18을 참조하면, 계층적 분할 구조에서 코딩 루틴(또는 스레드)들이 여러 비트 스트림에 코딩될 수 있다.

[251] 인코더/디코더는 코딩 쿼드 트리(coding_quadtree) 신택스를 코딩한다(S1801). 이때, 인코더/디코더는 코딩 쿼드 트리 신택스에서 CU 스레드들을 생성하여 CU 스레드 큐에 등록할 수 있다.

[252] 인코더/디코더는 CU 스레드 큐에 등록된 CU 스레드들을 코딩한다(S1802). 이때, 인코더/디코더는 상기 CU 스레드들을 병렬적으로(또는 동시에) 여러 비트 스트림에서 코딩할 수 있다. 또한, 인코더/디코더는 CU 신택스에서 PU 스레드들을 생성하여 PU 스레드 큐에 등록할 수 있고(앞서 도 11 참조), CU 신택스에서 변환 트리 스레드들을 생성하여 변환 트리 스레드 큐에 등록할 수 있다(앞서 도 16 참조).

[253] 인코더/디코더는 PU 스레드 큐에 등록된 PU 스레드들을 코딩한다(S1803). 이때, 인코더/디코더는 상기 PU 스레드들을 병렬적으로(또는 동시에) 여러 비트 스트림에서 코딩할 수 있다.

[254] 인코더/디코더는 변환 트리 스레드들을 코딩하고(S1804), TU 스레드들을 코딩한다(S1805). S1804 단계의 변환 트리와 S1805 단계의 TU는 순차적으로 동작하기 때문에, S1804 단계에서 변환 트리 스레드를 코딩하면서, TU 스레드(또는 TU 루틴)을 수행할 수 있다.

[255] 인코더/디코더는 레지듀얼 코딩(residual_coding) 신택스를 코딩한다(S1806).

[256] 인코더/디코더는 CG 스레드들을 코딩한다(S1807).

[257] 여기서, CG_i(즉, CG_1, CG_2, … ,CG_N)는 앞서 도 7에서 설명한 제 2 내지 제 7 서브 루틴(또는 스레드)를 나타낸다.

[258] CTU의 하위 계층(또는 기능 블록)에 대한 스레드를 모두 등록하고 수행하면, 구현에 따라 버퍼(buffer)의 요구량과 다른 기능 블록들로 데이터를 전달하기까지의 레이턴시(latency)가 길어질 수 있다.

[259] 따라서, 다른 기능 블록들에 대한 코딩 시작 시간을 앞당기기 위해(즉,

레이턴시를 줄이기 위해) 인코더/디코더는 생성되는 스레드들을 라운드 로빈(round-robin) 방식으로 비트 스트림에 할당하고, 코딩을 수행할 수 있다.

[260] 예를 들어, 4개의 비트 스트림에 스레드들을 라운드 로빈(round-robin) 방식으로 할당하는 경우를 가정하면, 인코더/디코더는 S1802 단계에서 4개의 스레드를 수행한 후, S1803 단계에서 4개의 스레드를 수행하여 움직임 보상을 수행할 수 있다.

[261] 이와 같은 방법을 적용하여, 움직임 보상(motion compensation) 또는 변환 및 양자화(transform and quantization) 모듈의 인코딩/디코딩 수행 시작까지의 레이턴시를 줄일 수 있다.

[262] 본 발명에서 제안하는 실시예들은 독립적으로 적용될 수도 있고, 복수 개의 실시예들이 조합되어 적용될 수도 있다.

[263] 도 19는 본 발명이 적용될 수 있는 실시예로서, 여러 비트 스트림으로 구성된 비디오 신호에 대해 엔트로피 디코딩을 수행하는 과정을 설명하는 흐름도이다.

[264] 디코더는 복수 개의 비트 스트림(bit stream)으로 구성된 비디오 신호를 수신할 수 있다(S1901).

[265] 디코더는 상기 비디오 신호의 각 비트 스트림 별로 신택스 요소(syntax element)의 심볼(symbol)을 복호화할 수 있다(S1902).

[266] 앞서 도 7, 도 9, 도 10 내지 12에서 설명한 바와 같이 상기 신택스 요소는 현재 블록 내에서 하나 이상의 신택스 요소가 독립적으로 처리되는 영역을 나타내는 기능 블록 단위(function block unit)로 상기 복수 개의 비트 스트림에 번갈아가며 매핑될 수 있다.

[267] 또한, 앞서 도 7, 도 8, 도 13 내지 17에서 설명한 바와 같이, 디코더는 기능 블록 단위 또는 신택스 요소 단위로 복수 개의 비트 스트림간 동기화를 수행할 수 있다.

[268] 전술한 바와 같이, 상기 동기화를 수행하기 위하여 디코더는 현재 비트 스트림에서 현재 신택스 요소의 복호화를 시작하기 위한 락(lock)을 획득하고, 상기 현재 신택스 요소의 복호화가 종료된 경우 상기 획득된 락을 반환할 수 있다. 디코더는 해당 비트 스트림에서 락이 획득된 경우에만 상기 현재 신택스 요소를 복호화할 수 있다.

[269] 또한, 디코더는 현재 비트 스트림 이외의 비트 스트림에서 상기 현재 신택스 요소에 대한 락을 획득하였는지 여부를 나타내는 락 상태 정보(lock state information)를 확인할 수 있다.

[270] 예를 들어, 상기 락 상태 정보는 컨텍스트 스토어(context store) 등 별도의 저장 공간에 저장될 수 있다. 현재 비트 스트림에서 현재 신택스 요소의 복호화를 위해 디코더는 락 요청 정보를 상기 컨텍스트 스토어(또는 메모리)에 전송하여 해당 락에 대한 획득을 시도할 수 있다. 또는 디코더는 상기 컨텍스트 스토어로부터 해당 락에 대한 상태 정보를 읽어서 다른 비트 스트림에서의 락의 획득 여부 또는 락의 소유권 정보를 확인할 수도 있다. 이 경우, 신택스 요소에

대한 복호화가 종료된 후 락 해제 정보와 함께 락 해제 요청을 상기 컨텍스트 스토어에 전송하여 다른 비트 스트림에서 해당 락에 관련된 컨텍스트들을 갱신할 수 있도록 허용할 수 있다.

[271] 또한, 전술한 바와 같이, 현재 신택스 요소에 컨텍스트(context)를 참조하는 정규 코딩(regular coding)이 적용되는 경우 디코더는 상기 복수 개의 비트 스트림간 동기화를 수행할 수 있고, 현재 신택스 요소에 바이패스 모드(bypass mode)가 적용되는 경우에는 상기 복수 개의 비트 스트림간 동기화를 수행하지 않을 수 있다.

[272] 또한, 앞서 도 9 및 도 12에서 설명한 바와 같이, 디코더는 기능 블록 단위로 신택스 요소의 심볼을 복호화한 후, 현재 비트 스트림을 전환할 수도 있다.

[273] 또한, 앞서, 도 10 및 도 13, 도 18에서 설명한 바와 같이, 디코더는 기능 블록 단위 마다 독립적으로 처리되는 작업 단위를 나타내는 스레드를 생성하여 상기 스레드의 정보를 저장하는 스레드 큐(thread queue)에 등록할 수 있고, 상기 스레드 큐에 저장된 복수 개의 스레드들을 병렬적으로 복호화할 수 있다.

[274] 디코더는 상기 복호화된 심볼을 이용하여 상기 신택스 요소의 정보를 도출할 수 있다(S1903).

[275] 디코더는 앞서 도 4 및 도 6에서 설명한 방법에 의해 디코딩된 이진수 형태의 빈을 입력 받아 신택스 요소의 값을 출력할 수 있다.

[276] 본 발명에서 어떠한 단위로 동기화하여 파이프라이닝을 수행할지 여부는 구현 의존적하다. 예를 들어, 하드웨어 구현에서는 신택스 요소 단위로 정교하게 동기화하는 것이 처리량 측면에서 유리할 수 있다. 그리고, 소프트웨어 구현에서는 OS에서 제공하는 동기화 프리미티브(synchronization primitive)(예를 들어, mutex lock, semaphore 등)의 성능에 따라 달라질 수 있으며, 성능이 더 좋을수록 보다 작은 단위로 동기화하는 것이 타당할 것이다.

[277]

[278] 상기 기술된 것과 같이, 본 발명에서 설명한 실시예들은 프로세서, 마이크로 프로세서, 컨트롤러 또는 칩 상에서 구현되어 수행될 수 있다. 예를 들어, 상기 도 1 내지 4 에서 도시한 기능 유닛들은 컴퓨터, 프로세서, 마이크로 프로세서, 컨트롤러 또는 칩 상에서 구현되어 수행될 수 있다.

[279] 또한, 본 발명이 적용되는 디코더 및 인코더는 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, 인터넷 스트리밍 서비스 제공 장치, 3차원(3D) 비디오 장치, 화상 전화 비디오 장치, 및 의료용 비디오 장치 등에 포함될 수 있으며, 비디오 신호 및 데이터 신호를 처리하기 위해 사용될 수 있다.

[280] 또한, 본 발명이 적용되는 처리 방법은 컴퓨터로 실행되는 프로그램의 형태로 생산될 수 있으며, 컴퓨터가 판독할 수 있는 기록 매체에 저장될 수 있다. 본

발명에 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 장치를 포함한다. 상기 컴퓨터가 판독할 수 있는 기록 매체는, 예를 들어, 블루레이 디스크(BD), 범용 직렬 버스(USB), ROM, 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 비트 스트림이 컴퓨터가 판독할 수 있는 기록 매체에 저장되거나 유무선 통신 네트워크를 통해 전송될 수 있다.

[281]

산업상 이용가능성

[282] 이상, 전술한 본 발명의 바람직한 실시예는, 예시의 목적을 위해 개시된 것으로, 당업자라면 이하 첨부된 특허청구범위에 개시된 본 발명의 기술적 사상과 그 기술적 범위 내에서, 다양한 다른 실시예들을 개량, 변경, 대체 또는 부가 등이 가능할 것이다.

청구범위

[청구항 1] 비디오 신호에 대해 엔트로피 디코딩을 수행하는 방법에 있어서,
복수 개의 비트 스트림(bit stream)으로 구성된 비디오 신호를 수신하는 단계;
상기 비디오 신호의 각 비트 스트림 별로 신택스 요소(syntax element)의 심볼(symbol)을 복호화하는 단계; 및
상기 복호화된 심볼을 이용하여 상기 신택스 요소의 정보를 도출하는 단계를 포함하되,
상기 신택스 요소는 현재 블록 내에서 하나 이상의 신택스 요소가 독립적으로 처리되는 영역을 나타내는 기능 블록 단위(function block unit)로 상기 복수 개의 비트 스트림에 번갈아가며 매핑되는 방법.

[청구항 2] 제 1항에 있어서,
상기 신택스 요소의 심볼을 복호화하는 단계는,
상기 기능 블록 단위로 상기 복수 개의 비트 스트림간 동기화를 수행하는 단계를 더 포함하는 방법.

[청구항 3] 제 1항에 있어서,
상기 신택스 요소의 심볼을 복호화하는 단계는,
상기 신택스 요소 단위로 상기 복수 개의 비트 스트림간 동기화를 수행하는 단계를 더 포함하는 방법.

[청구항 4] 제 3항에 있어서,
상기 복수 개의 비트 스트림간 동기화를 수행하는 단계는,
현재 비트 스트림에서 현재 신택스 요소의 복호화를 시작하기 위한 락(lock)을 획득하는 단계를 더 포함하고,
상기 락을 획득한 경우에만 상기 현재 신택스 요소의 심볼을 복호화하는 방법.

[청구항 5] 제 4항에 있어서,
상기 복수 개의 비트 스트림간 동기화를 수행하는 단계는,
상기 현재 신택스 요소의 복호화가 종료된 경우 상기 획득된 락을 반환하는 단계를 더 포함하는 방법.

[청구항 6] 제 4항에 있어서,
상기 복수 개의 비트 스트림간 동기화를 수행하는 단계는,
상기 현재 비트 스트림 이외의 비트 스트림에서 상기 현재 신택스 요소에 대한 락을 획득하였는지 여부를 나타내는 락 상태 정보(lock state information)를 확인하는 단계를 더 포함하는 방법.

[청구항 7] 제 3항에 있어서,
상기 복수 개의 비트 스트림간 동기화를 수행하는 단계는,
현재 신택스 요소에 컨텍스트(context)를 참조하는 정규 코딩(regular

coding)이 적용되는 경우, 상기 복수 개의 비트 스트림간 동기화를 수행하는 방법.

[청구항 8] 제 1항에 있어서,
상기 신택스 요소의 심볼을 복호화하는 단계는,
상기 기능 블록 단위로 신택스 요소의 심볼을 복호화한 후 현재 비트 스트림을 전환하는 단계를 더 포함하는 방법.

[청구항 9] 제 1항에 있어서,
상기 기능 블록 단위 마다 독립적으로 처리되는 작업 단위를 나타내는 스레드(thread)를 생성하여 상기 스레드의 정보를 저장하는 스레드 큐(thread queue)에 등록하는 단계; 및
상기 스레드 큐에 저장된 복수 개의 스레드들을 병렬적으로 복호화하는 단계를 더 포함하는 방법.

[청구항 10] 비디오 신호에 대해 엔트로피 디코딩을 수행하는 장치에 있어서,
복수 개의 비트 스트림(bit stream)으로 구성된 비디오 신호를 수신하는 비디오 신호 수신부;
상기 비디오 신호의 각 비트 스트림 별로 신택스 요소(syntax element)의 심볼(symbol)을 복호화하는 심볼 복호화부; 및
상기 복호화된 심볼을 이용하여 상기 신택스 요소의 정보를 복호화하는 신택스 요소 복호화부를 포함하되,
상기 신택스 요소는 현재 블록 내에서 하나 이상의 신택스 요소가 독립적으로 처리되는 영역을 나타내는 기능 블록 단위(function block unit)로 상기 복수 개의 비트 스트림에 번갈아가며 매핑되는 장치.

[청구항 11] 제 10항에 있어서,
상기 심볼 복호화부는,
상기 기능 블록 단위로 상기 복수 개의 비트 스트림간 동기화를 수행하는 동기화부를 포함하는 장치.

[청구항 12] 제 10항에 있어서,
상기 심볼 복호화부는,
상기 신택스 요소 단위로 상기 복수 개의 비트 스트림간 동기화를 수행하는 동기화부를 포함하는 장치.

[청구항 13] 제 12항에 있어서,
상기 동기화부는,
현재 비트 스트림에서 현재 신택스 요소의 복호화를 시작하기 위한 락(lock)을 획득하고,
상기 락이 획득된 경우에만 상기 현재 신택스 요소의 심볼이 복호화되는 장치.

[청구항 14] 제 13항에 있어서,
상기 동기화부는,

상기 현재 신택스 요소의 복호화가 종료된 경우 상기 획득된 락을 반환하는 장치.

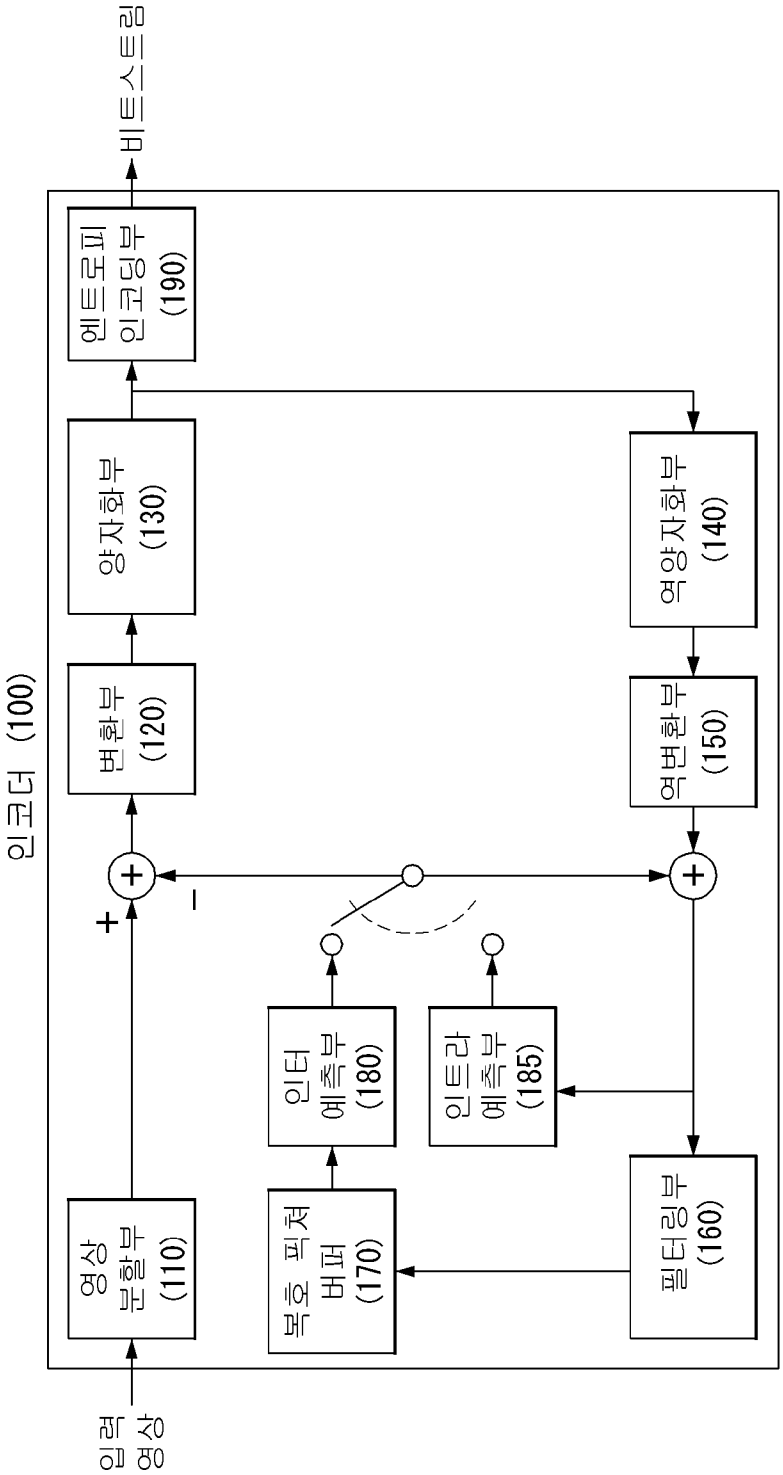
[청구항 15] 제 13항에 있어서,
상기 동기화부는,
상기 현재 비트 스트림 이외의 비트 스트림에서 상기 현재 신택스 요소에 대한 락을 획득하였는지 여부를 나타내는 락 상태 정보(lock state information)를 확인하는 장치.

[청구항 16] 제 12항에 있어서,
상기 동기화부는,
현재 신택스 요소에 컨텍스트(context)를 참조하는 정규 코딩(regular coding)이 적용되는 경우, 상기 복수 개의 비트 스트림간 동기화를 수행하는 장치.

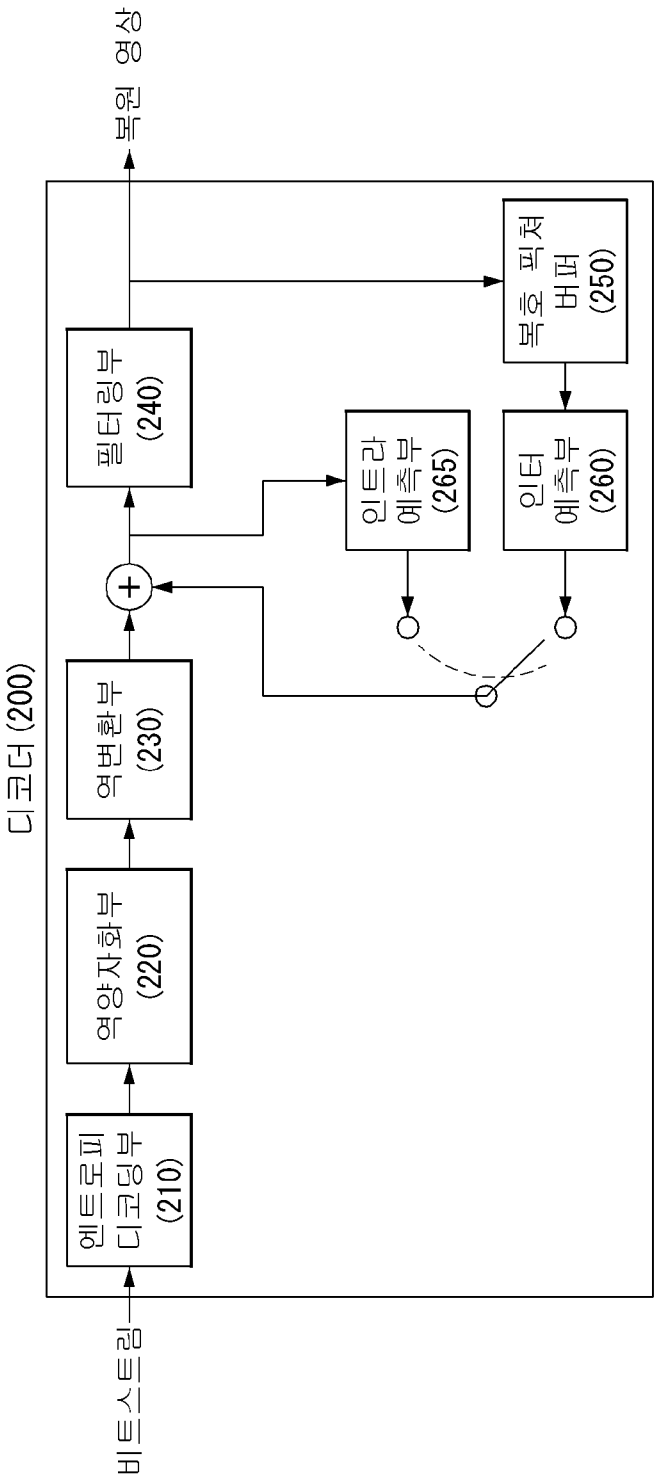
[청구항 17] 제 10항에 있어서,
상기 심볼 복호화부는,
상기 기능 블록 단위로 신택스 요소의 심볼을 복호화한 후 현재 비트 스트림을 전환하는 장치.

[청구항 18] 제 10항에 있어서,
상기 기능 블록 단위 마다 독립적으로 처리되는 작업 단위를 나타내는 스레드(thread)를 생성하여 상기 스레드의 정보를 저장하는 스레드 큐(thread queue)에 등록하는 스레드 생성부; 및
상기 스레드 큐에 저장된 복수 개의 스레드들을 병렬적으로 복호화하는 스레드 복호화부를 더 포함하는 장치.

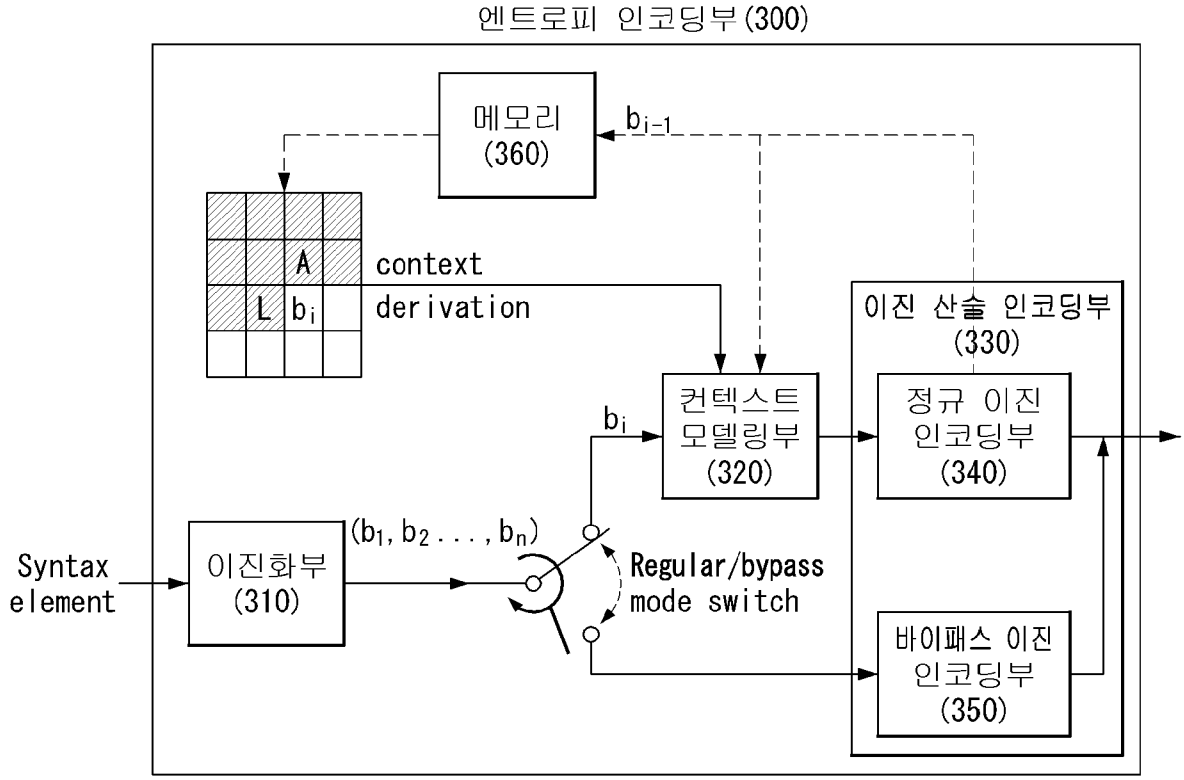
[도1]


비트스트림
엔트로피 인코딩부 (190)
양자화부 (130)
변환부 (120)
영상 분할부 (110)
입력 영상
인코더 (100)
역양자화부 (140)
역변환부 (150)
필터링부 (160)
복호 픽쳐 버퍼 (170)
인터 예측부 (180)
인트라 예측부 (185)
+
-

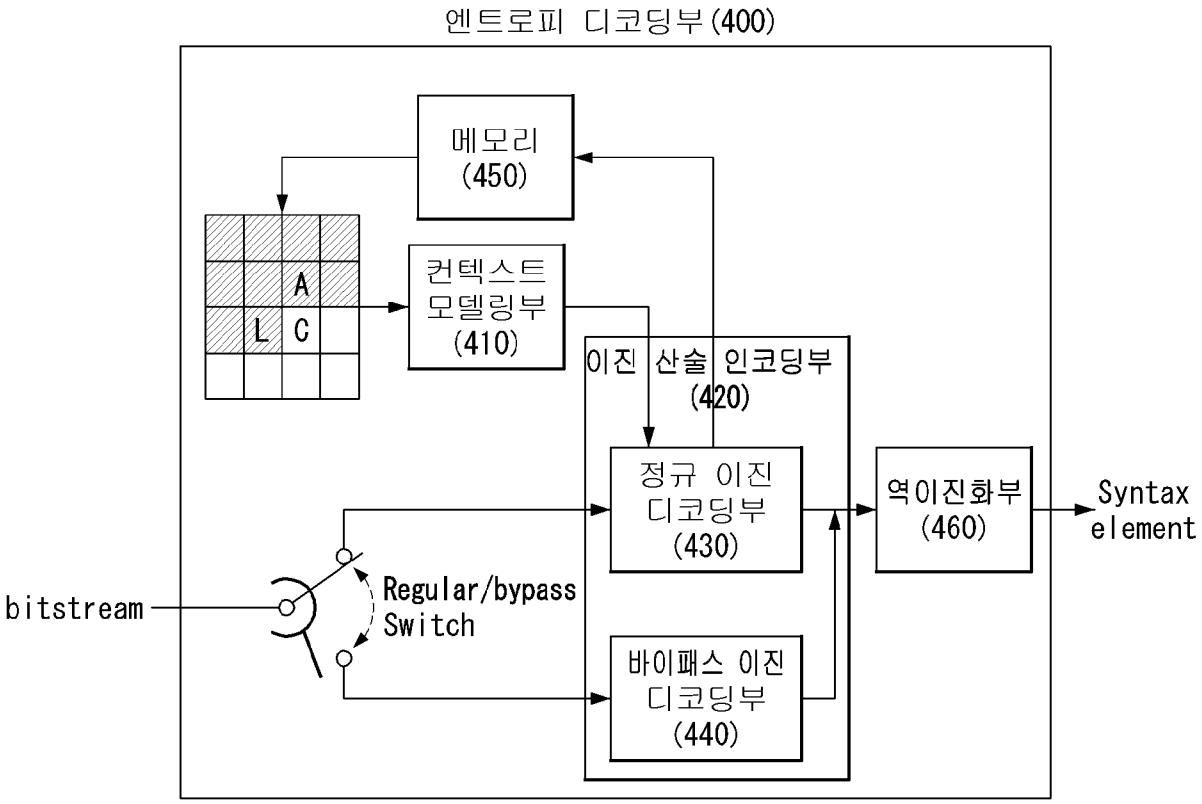
[도2]


비트스트림
엔트로피 디코딩부 (210)
역양자화부 (220)
역변환부 (230)
필터링부 (240)
복호 픽처 버퍼 (250)
인트라 예측부 (265)
인터 예측부 (260)
디코더 (200)
복원 영상

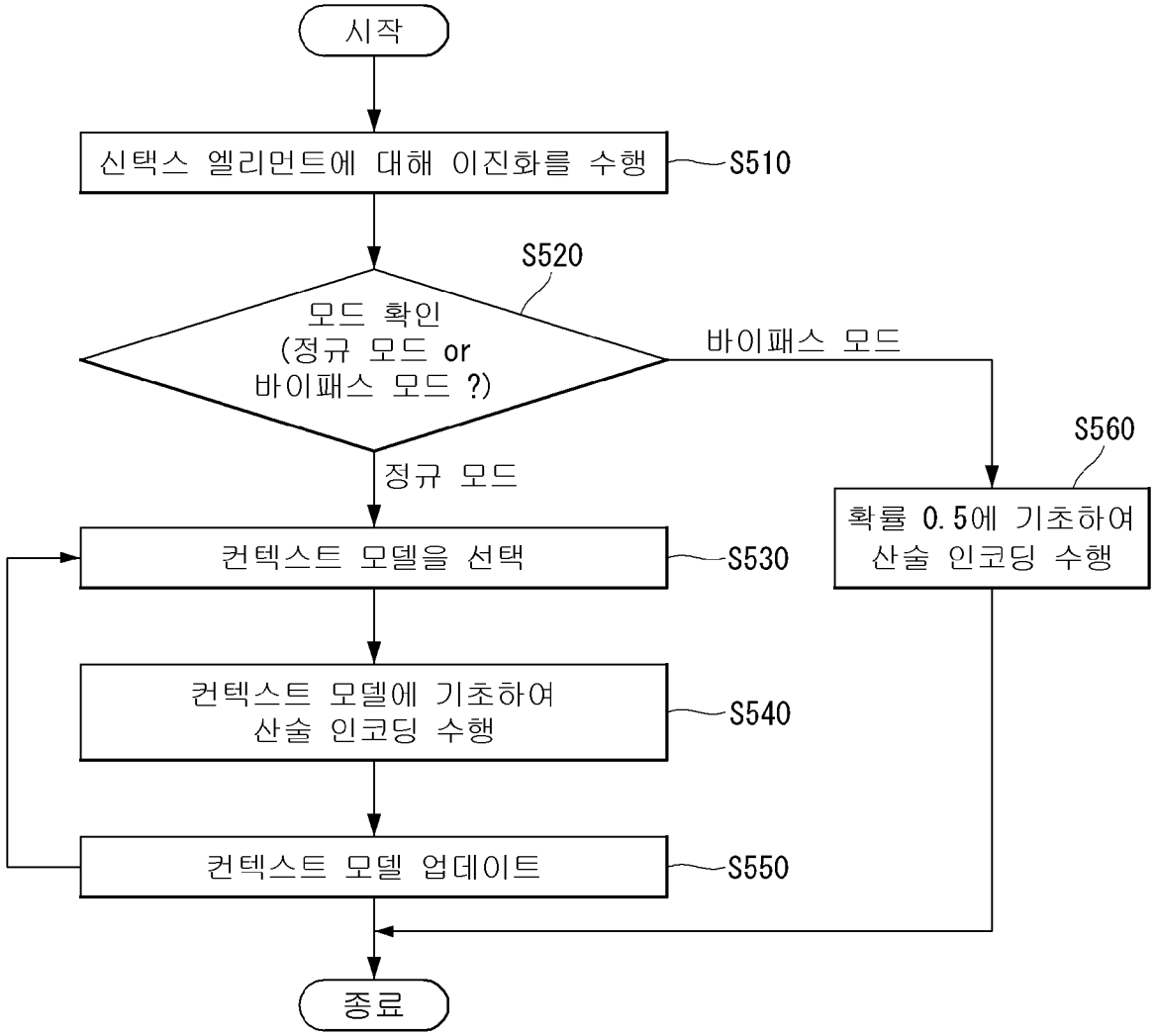
[도3]


엔트로피 인코딩부(300)
메모리
(360)
b_{i-1}
A
L b_i
context
derivation
이진 산술 인코딩부
(330)
컨텍스트
모델링부
(320)
정규 이진
인코딩부
(340)
b_i
Syntax
element
이진화부
(310)
$(b_1, b_2 \ldots, b_n)$
Regular/bypass
mode switch
바이패스 이진
인코딩부
(350)

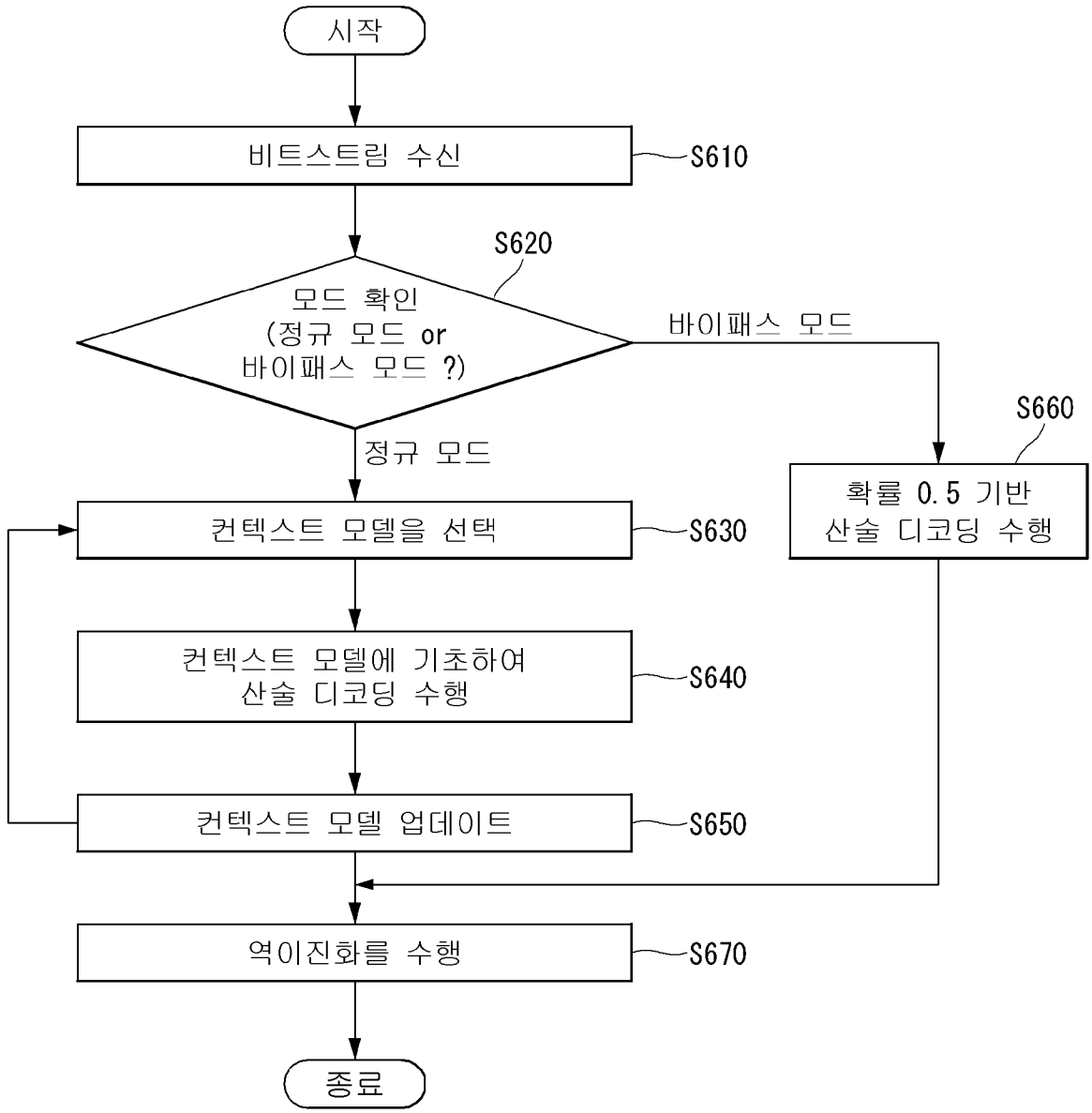
[도4]


엔트로피 디코딩부(400)
메모리
(450)
A
L C
컨텍스트
모델링부
(410)
이진 산술 인코딩부
(420)
정규 이진
디코딩부
(430)
역이진화부
(460)
Syntax
element
bitstream
Regular/bypass
Switch
바이패스 이진
디코딩부
(440)

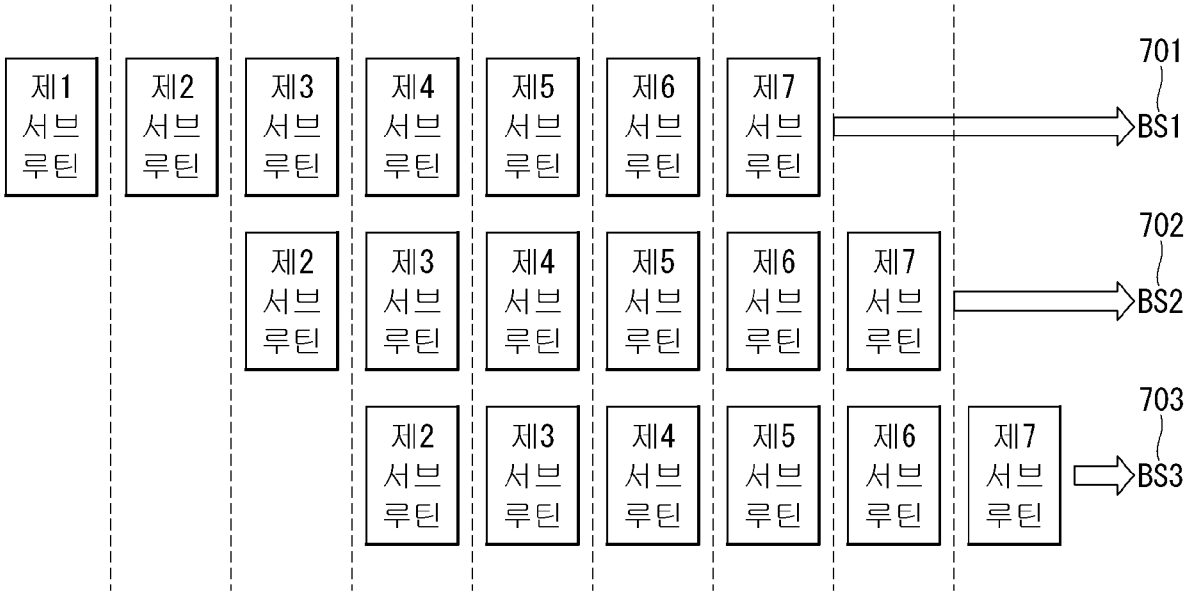
[도5]


시작
신택스 엘리먼트에 대해 이진화를 수행
S510
S520
모드 확인
(정규 모드 or
바이패스 모드 ?)
바이패스 모드
정규 모드
S560
확률 0.5에 기초하여
산술 인코딩 수행
컨텍스트 모델을 선택
S530
컨텍스트 모델에 기초하여
산술 인코딩 수행
S540
컨텍스트 모델 업데이트
S550
종료

[도6]


시작
비트스트림 수신
S610
S620
모드 확인
(정규 모드 or
바이패스 모드 ?)
바이패스 모드
정규 모드
S660
확률 0.5 기반
산술 디코딩 수행
컨텍스트 모델을 선택
S630
컨텍스트 모델에 기초하여
산술 디코딩 수행
S640
컨텍스트 모델 업데이트
S650
역이진화를 수행
S670
종료

[도7]


제1 서브 루틴
제2 서브 루틴
제3 서브 루틴
제4 서브 루틴
제5 서브 루틴
제6 서브 루틴
제7 서브 루틴
701
BS1
제2 서브 루틴
제3 서브 루틴
제4 서브 루틴
제5 서브 루틴
제6 서브 루틴
제7 서브 루틴
702
BS2
제2 서브 루틴
제3 서브 루틴
제4 서브 루틴
제5 서브 루틴
제6 서브 루틴
제7 서브 루틴
703
BS3

[도8a]

residual_coding(x0, y0, log2TrafoSize, cIdx) {	Descriptor
if(transform_skip_enabled_flag && !cu_transquant_bypass_flag && (log2TrafoSize == 2))	
transform_skip_flag[x0][y0][cIdx]	ae(v)
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)
lastScanPos = 16	
lastSubBlock = (1 << (log2TrafoSize - 2)) * (1 << (log2TrafoSize - 2)) - 1	
do {	
if(lastScanPos == 0) {	
lastScanPos = 16	
lastSubBlock- -	
}	
lastScanPos- -	
xS = ScanOrder[log2TrafoSize - 2][scanIdx][lastSubBlock][0]	
yS = ScanOrder[log2TrafoSize - 2][scanIdx][lastSubBlock][1]	
xC = (xS << 2) + ScanOrder[2][scanIdx][lastScanPos][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][lastScanPos][1]	
} while((xC != LastSignificantCoeffX) || (yC != LastSignificantCoeffY))	
for(i = lastSubBlock; i >= 0; i- -) {	
xS = ScanOrder[log2TrafoSize - 2][scanIdx][i][0]	
yS = ScanOrder[log2TrafoSize - 2][scanIdx][i][1]	
inferSbDcSigCoeffFlag = 0	
if((i < lastSubBlock) && (i > 0)) {	
S801 (LA)coded_sub_block_flag[xS][yS](LR)	ae(v)
inferSbDcSigCoeffFlag = 1	
}	

[도8b]

S802

(LA)	
for(n = (i == lastSubBlock) ? lastScanPos - 1 : 15; n >= 0; n- -) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(coded_sub_block_flag[xS][yS] && (n > 0 || !inferSbDcSigCoeffFlag)) {	
sig_coeff_flag[xC][yC]	ae(v)
if(sig_coeff_flag[xC][yC])	
inferSbDcSigCoeffFlag = 0	
}	
}	
(LR)	
firstSigScanPos = 16	
lastSigScanPos = -1	
numGreater1Flag = 0	
lastGreater1ScanPos = -1	
(LA)	
for(n = 15; n >= 0; n- -) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC]) {	
if(numGreater1Flag < 8) {	
coeff_abs_level_greater1_flag[n]	ae(v)
numGreater1Flag++	
if(coeff_abs_level_greater1_flag[n] && lastGreater1ScanPos == -1)	
lastGreater1ScanPos = n	
}	
if(lastSigScanPos == -1)	
lastSigScanPos = n	
firstSigScanPos = n	
}	
}	
(LR)	

[도8c]

signHidden = (lastSigScanPos - firstSigScanPos > 3 && !cu_transquant_bypass_flag)	
if(lastGreater1ScanPos != -1)	
(LA) coeff_abs_level_greater2_flag[lastGreater1ScanPos](LR)	ae(v)
for(n = 15; n >= 0; n- -) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC] && (!sign_data_hiding_ enabled_flag || !signHidden || (n != firstSigScanPos)))	
S803 coeff_sign_flag[n]	ae(v)
}	
numSigCoeff = 0	
sumAbsLevel = 0	
for(n = 15; n >= 0; n- -) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC]) {	
baseLevel = 1 + coeff_abs_level_greater1_flag[n] + coeff_abs_level_greater2_flag[n]	
if(baseLevel == ((numSigCoeff < 8) ? ((n == lastGreater1ScanPos) ? 3 : 2) : 1))	
S804 coeff_abs_level_remaining[n]	ae(v)
TransCoeffLevel[x0][y0][cIdx][xC][yC] = (coeff_abs_level_remaining[n] + baseLevel) * (1 - 2 * coeff_sign_flag[n])	
if(sign_data_hiding_enabled_flag && signHidden) {	
sumAbsLevel += (coeff_abs_level_remaining[n] + baseLevel)	
if((n == firstSigScanPos) && ((sumAbsLevel % 2) == 1))	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = -TransCoeffLevel[x0][y0][cIdx][xC][yC]	
}	
numSigCoeff++	
}	
}	
}	
}	

[도9]

	residual_coding(x0, y0, log2TrafoSize, cIdx) {	Descriptor
	.....	
	for(i = lastSubBlock; i >= 0; i--) {	
	.....	
S901	set_next_bitstream ( )	
	}	
	}	

[도10]

	coding_quadtree(x0, y0, log2CbSize, cqtDepth) {	Descriptor
	if(x0 + (1 << log2CbSize) <= pic_width_in_luma_samples && y0 + (1 << log2CbSize) <= pic_height_in_luma_samples && log2CbSize > MinCbLog2SizeY)	
S1001	split_cu_flag[x0][y0]	ae(v)
	if(cu_qp_delta_enabled_flag && log2CbSize >= Log2MinCuQpDeltaSize) {	
	IsCuQpDeltaCoded = 0	
	CuQpDeltaVal = 0	
	}	
	if(split_cu_flag[x0][y0]) {	
	x1 = x0 + (1 << (log2CbSize - 1))	
	y1 = y0 + (1 << (log2CbSize - 1))	
	coding_quadtree(x0, y0, log2CbSize - 1, cqtDepth + 1)	
	if(x1 < pic_width_in_luma_samples)	
	coding_quadtree(x1, y0, log2CbSize - 1, cqtDepth + 1)	
	if(y1 < pic_height_in_luma_samples)	
	coding_quadtree(x0, y1, log2CbSize - 1, cqtDepth + 1)	
	if(x1 < pic_width_in_luma_samples && y1 < pic_height_in_luma_samples)	
	coding_quadtree(x1, y1, log2CbSize - 1, cqtDepth + 1)	
	} else	
S1002	issue_coding_unit_thread(x0, y0, log2CbSize)	
	}	

[도11]

	coding_unit(x0, y0, log2CbSize) {	Descriptor
	.....	
	if(cu_skip_flag[x0][y0])	
S1101	issue_prediction_unit_thread(x0, y0, nCbS, nCbS)	
	else {	
	.....	
	if(CuPredMode[x0][y0] == MODE_INTRA) {	
	.....	
	} else {	
	if(PartMode == PART_2Nx2N)	
S1102	merge_flag[x0][y0]	
S1103	issue_prediction_unit_thread (x0, y0, nCbS, nCbS)	
	else if(PartMode == PART_2NxN) {	
	issue_prediction_unit_thread (x0, y0, nCbS, nCbS / 2)	
	issue_prediction_unit_thread (x0, y0 + (nCbS / 2), nCbS, nCbS / 2)	
	} else if(PartMode == PART_Nx2N) {	
	issue_prediction_unit_thread (x0, y0, nCbS / 2, nCbS)	
	issue_prediction_unit_thread (x0 + (nCbS / 2), y0, nCbS / 2, nCbS)	
	} else if(PartMode == PART_2NxnU) {	
	issue_prediction_unit_thread (x0, y0, nCbS, nCbS / 4)	
	issue_prediction_unit_thread (x0, y0 + (nCbS / 4), nCbS, nCbS * 3 / 4)	
	} else if(PartMode == PART_2NxnD) {	
	issue_prediction_unit_thread (x0, y0, nCbS, nCbS * 3 / 4)	
	issue_prediction_unit_thread (x0, y0 + (nCbS * 3 / 4), nCbS, nCbS / 4)	
	} else if(PartMode == PART_nLx2N) {	
	issue_prediction_unit_thread (x0, y0, nCbS / 4, nCbS)	
	issue_prediction_unit_thread (x0 + (nCbS / 4), y0, nCbS * 3 / 4, nCbS)	
	} else if(PartMode == PART_nRx2N) {	
	issue_prediction_unit_thread (x0, y0, nCbS * 3 / 4, nCbS)	
	issue_prediction_unit_thread (x0 + (nCbS * 3 / 4), y0, nCbS / 4, nCbS)	
	} else { /* PART_NxN */	
	issue_prediction_unit_thread (x0, y0, nCbS / 2, nCbS / 2)	
	issue_prediction_unit_thread (x0 + (nCbS / 2), y0, nCbS / 2, nCbS / 2)	
	issue_prediction_unit_thread (x0, y0 + (nCbS / 2), nCbS / 2, nCbS / 2)	
	issue_prediction_unit_thread (x0 + (nCbS / 2), y0 + (nCbS / 2), nCbS / 2, nCbS / 2)	
	}	
	}	
	.....	
	}	
	}	

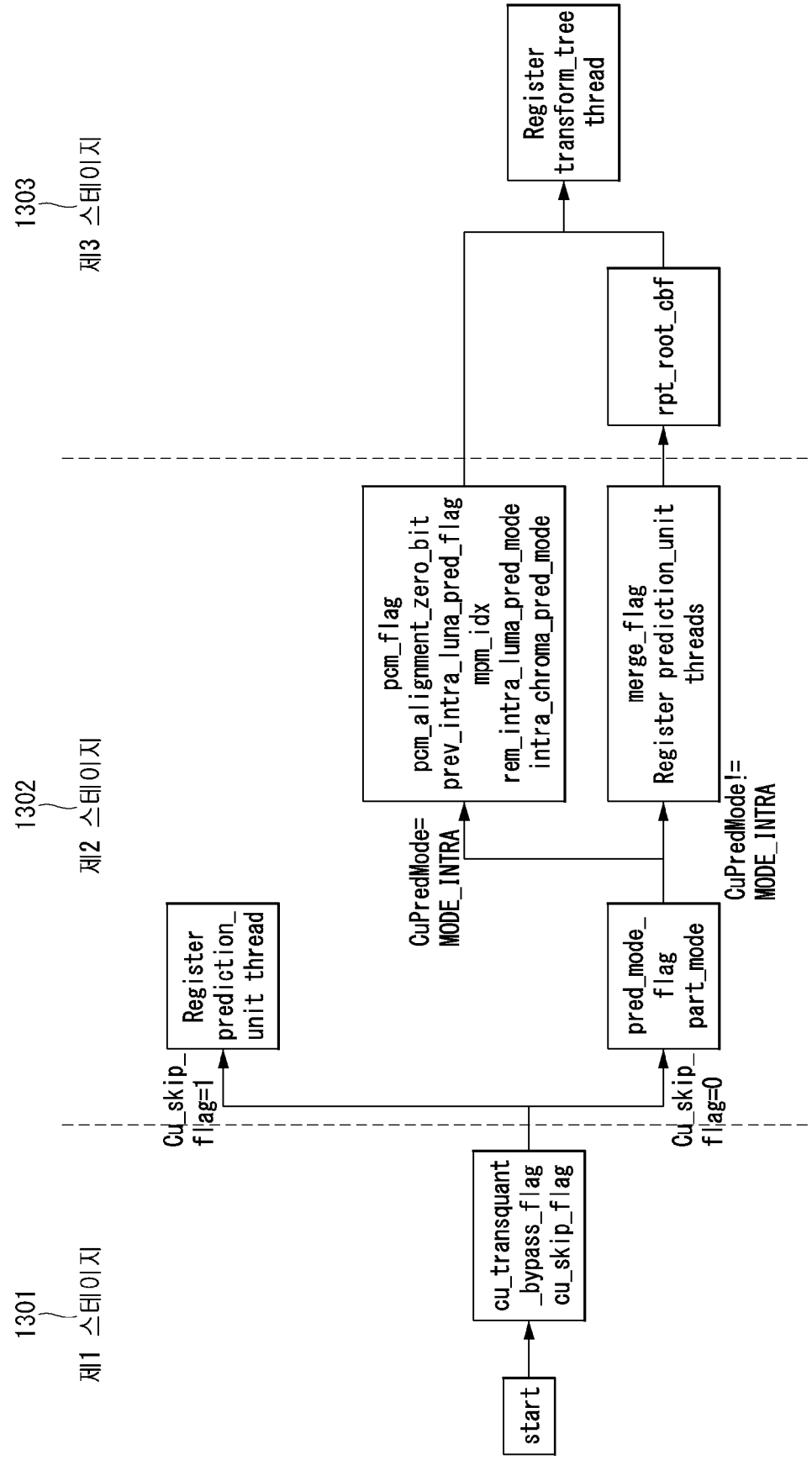
[도12a]

	coding_unit(x0, y0, log2CbSize) {	Descriptor
	.....	
	if(cu_skip_flag[x0][y0]) [	
	prediction_unit(x0, y0, nCbS, nCbS)	
S1201	set_next_bitstream( )	
	else {	
	.....	
	if(CuPredMode[x0][y0] == MODE_INTRA) {	
	.....	
	} else {	
	if(PartMode == PART_2Nx2N)	
	prediction_unit(x0, y0, nCbS, nCbS)	
	set_next_bitstream( )	
	else if(PartMode == PART_2NxN) {	
	prediction_unit(x0, y0, nCbS, nCbS / 2)	
S1202	set_next_bitstream( )	
	prediction_unit(x0, y0 + (nCbS / 2), nCbS, nCbS / 2)	
	set_next_bitstream( )	
	} else if(PartMode == PART_Nx2N) {	
	prediction_unit(x0, y0, nCbS / 2, nCbS)	
	set_next_bitstream( )	
	prediction_unit(x0 + (nCbS / 2), y0, nCbS / 2, nCbS)	
	set_next_bitstream( )	
	} else if(PartMode == PART_2NxnU) {	
	prediction_unit(x0, y0, nCbS, nCbS / 4)	
	set_next_bitstream( )	
	prediction_unit(x0, y0 + (nCbS / 4), nCbS, nCbS * 3 / 4)	
	set_next_bitstream( )	
	} else if(PartMode == PART_2NxnD) {	
	prediction_unit(x0, y0, nCbS, nCbS * 3 / 4)	
	set_next_bitstream( )	
	prediction_unit(x0, y0 + (nCbS * 3 / 4), nCbS, nCbS / 4)	
	set_next_bitstream( )	

[도12b]

} else if(PartMode == PART_nLx2N) {	
prediction_unit(x0, y0, nCbS / 4, nCbS)	
set_next_bitstream( )	
prediction_unit(x0 + (nCbS / 4), y0, nCbS * 3 / 4, nCbS)	
set_next_bitstream( )	
} else if(PartMode == PART_nRx2N) {	
prediction_unit(x0, y0, nCbS * 3 / 4, nCbS)	
set_next_bitstream( )	
prediction_unit(x0 + (nCbS * 3 / 4), y0, nCbS / 4, nCbS)	
set_next_bitstream( )	
} else { /* PART_NxN */	
prediction_unit(x0, y0, nCbS / 2, nCbS / 2)	
set_next_bitstream( )	
prediction_unit(x0 + (nCbS / 2), y0, nCbS / 2, nCbS / 2)	
set_next_bitstream( )	
prediction_unit(x0, y0 + (nCbS / 2), nCbS / 2, nCbS / 2)	
set_next_bitstream( )	
prediction_unit(x0 + (nCbS / 2), y0 + (nCbS / 2), nCbS / 2, nCbS / 2)	
set_next_bitstream( )	
}	
}	
if(!pcm_flag[x0][y0]) {	
if(CuPredMode[x0][y0] != MODE_INTRA && !(PartMode == PART_2Nx2N && merge_flag[x0][y0]))	
rqt_root_cbf	ae(v)
if(rqt_root_cbf) {	
MaxTrafoDepth = (CuPredMode[x0][y0] == MODE_INTRA ? (max_transform_hierarchy_depth_intra + IntraSplitFlag) : max_transform_hierarchy_depth_inter)	
transform_tree(x0, y0, x0, y0, log2CbSize, 0, 0)	
}	
}	
}	
}	

[도13]


1301
제1 스테이지
1302
제2 스테이지
1303
제3 스테이지
start
cu_transquant_bypass_flag
cu_skip_flag
Cu_skip_flag=1
Cu_skip_flag=0
Register prediction_unit thread
CuPredMode= MODE_INTRA
CuPredMode!= MODE_INTRA
pcm_flag
pcm_alignment_zero_bit
prev_intra_luna_pred_flag
mpm_idx
rem_intra_luma_pred_mode
intra_chroma_pred_mode
pred_mode_flag
part_mode
merge_flag
Register prediction_unit threads
rpt_root_cbf
Register transform_tree thread

[도14a]

coding_unit(x0, y0, log2CbSize) {	Descriptor
if(transquant_bypass_enabled_flag)	
S1401 (LA)cu_transquant_bypass_flag(LR)	ae(v)
if(slice_type != I)	
(LA)cu_skip_flag[x0][y0](LR)	ae(v)
nCbS = (1 << log2CbSize)	
if(cu_skip_flag[x0][y0])	
(LA)issue_prediction_unit_thread(x0, y0, nCbS, nCbS)(LR)	
else {	
if(slice_type != I)	
(LA)pred_mode_flag(LR)	ae(v)
if(CuPredMode[x0][y0] != MODE_INTRA || log2CbSize == MinCbLog2SizeY)	
(LA)part_mode(LR)	ae(v)
if(CuPredMode[x0][y0] == MODE_INTRA) {	
if(PartMode == PART_2Nx2N && pcm_enabled_flag && log2CbSize >= Log2MinIpcmCbSizeY && log2CbSize <= Log2MaxIpcmCbSizeY)	
(LA)pcm_flag[x0][y0](LR)	ae(v)
if(pcm_flag[x0][y0]) {	
while(!byte_aligned( ))	
pcm_alignment_zero_bit	f(1)
pcm_sample(x0, y0, log2CbSize)	
} else {	
pbOffset = (PartMode == PART_NxN) ? (nCbS / 2) : nCbS	
(LA)	
for(j = 0; j < nCbS; j = j + pbOffset)	
for(i = 0; i < nCbS; i = i + pbOffset)	
prev_intra_luma_pred_flag[x0 + i][y0 + j]	ae(v)
(LR)	
for(j = 0; j < nCbS; j = j + pbOffset)	
for(i = 0; i < nCbS; i = i + pbOffset)	
if(prev_intra_luma_pred_flag[x0 + i][y0 + j])	
mpm_idx[x0 + i][y0 + j]	ae(v)
else	
rem_intra_luma_pred_mode[x0 + i][y0 + j]	ae(v)
(LA)intra_chroma_pred_mode[x0][y0](LR)	ae(v)
}	
} else {	

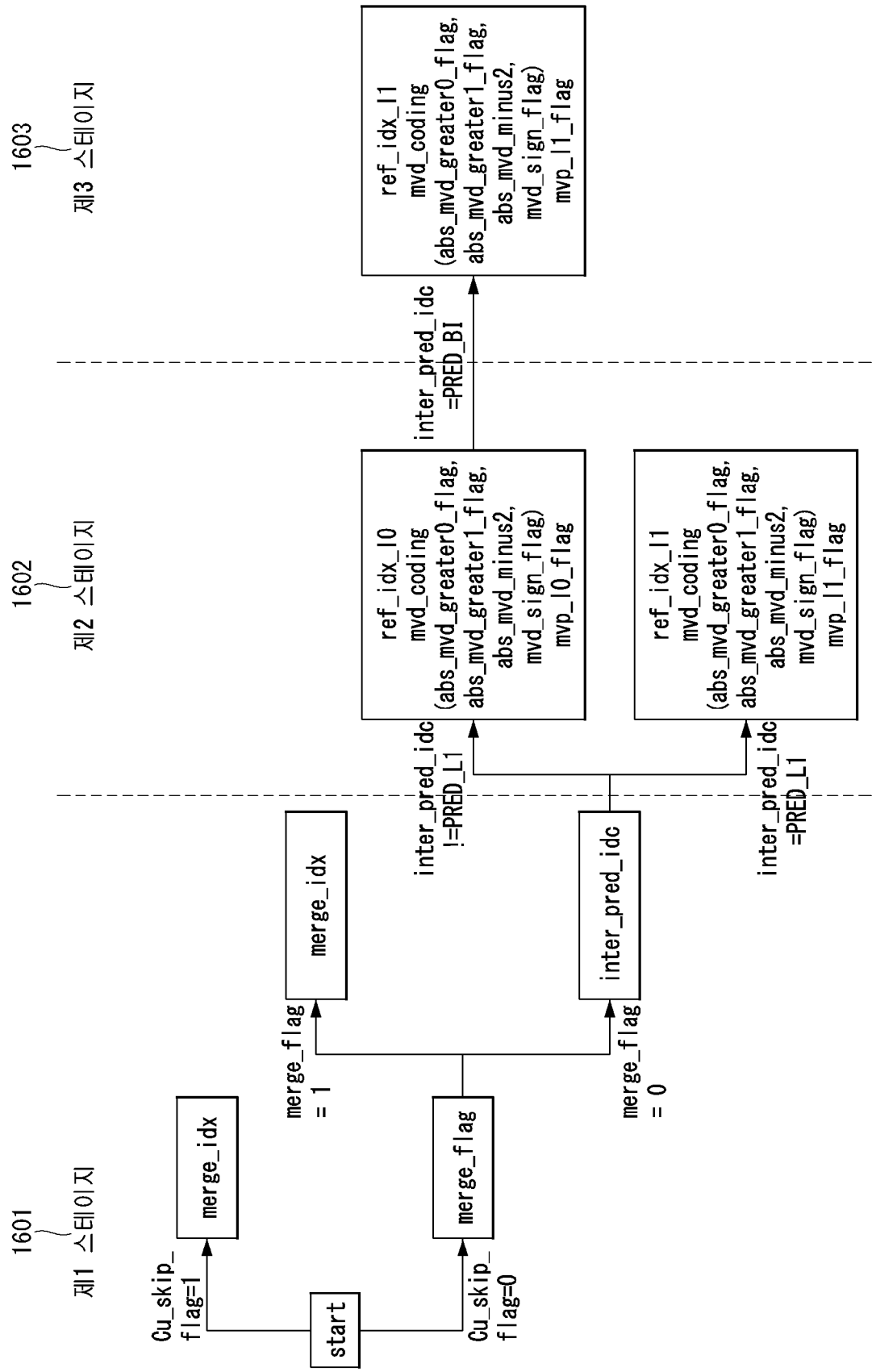
[도14b]

if(PartMode == PART_2Nx2N)	
S1402 (LA) merge_flag[x0][y0] (LR)	ae(v)
S1403 (LA)	
if(PartMode == PART_2Nx2N)	
issue_prediction_unit_thread (x0, y0, nCbS, nCbS)	
else if(PartMode == PART_2NxN) {	
issue_prediction_unit_thread (x0, y0, nCbS, nCbS / 2)	
issue_prediction_unit_thread (x0, y0 + (nCbS / 2), nCbS, nCbS / 2)	
} else if(PartMode == PART_Nx2N) {	
issue_prediction_unit_thread (x0, y0, nCbS / 2, nCbS)	
issue_prediction_unit_thread (x0 + (nCbS / 2), y0, nCbS / 2, nCbS)	
} else if(PartMode == PART_2NxnU) {	
issue_prediction_unit_thread (x0, y0, nCbS, nCbS / 4)	
issue_prediction_unit_thread (x0, y0 + (nCbS / 4), nCbS, nCbS * 3 / 4)	
} else if(PartMode == PART_2NxnD) {	
issue_prediction_unit_thread (x0, y0, nCbS, nCbS * 3 / 4)	
issue_prediction_unit_thread (x0, y0 + (nCbS * 3 / 4), nCbS, nCbS / 4)	
} else if(PartMode == PART_nLx2N) {	
issue_prediction_unit_thread (x0, y0, nCbS / 4, nCbS)	
issue_prediction_unit_thread (x0 + (nCbS / 4), y0, nCbS * 3 / 4, nCbS)	
} else if(PartMode == PART_nRx2N) {	
issue_prediction_unit_thread (x0, y0, nCbS * 3 / 4, nCbS)	
issue_prediction_unit_thread (x0 + (nCbS * 3 / 4), y0, nCbS / 4, nCbS)	
} else { /* PART_NxN */	
issue_prediction_unit_thread (x0, y0, nCbS / 2, nCbS / 2)	
issue_prediction_unit_thread (x0 + (nCbS / 2), y0, nCbS / 2, nCbS / 2)	
issue_prediction_unit_thread (x0, y0 + (nCbS / 2), nCbS / 2, nCbS / 2)	
issue_prediction_unit_thread (x0 + (nCbS / 2), y0 + (nCbS / 2), nCbS / 2, nCbS / 2)	
}	
(LR)	
}	
if(!pcm_flag[x0][y0]) {	
if(CuPredMode[x0][y0] != MODE_INTRA && !(PartMode == PART_2Nx2N && merge_flag[x0][y0]))	
(LA) rqt_root_cbf (LR)	ae(v)
if(rqt_root_cbf) {	
MaxTrafoDepth = (CuPredMode[x0][y0] == MODE_INTRA ? (max_transform_hierarchy_depth_intra + IntraSplitFlag) : max_transform_hierarchy_depth_inter)	
(LA) issue_transform_tree_thread(x0, y0, x0, y0, log2CbSize, 0, 0) (LR)	
}	
}	
}	
}	

[도15]

		Descriptor
	prediction_unit(x0, y0, nPbW, nPbH) {	
	if(cu_skip_flag[x0][y0]) {	
	if(MaxNumMergeCand > 1)	
	merge_idx[x0][y0]	ae(v)
	} else { /* MODE_INTER */	
S1501	if(PartMode != PART_2Nx2N)	
	merge_flag[x0][y0]	ae(v)
	if(merge_flag[x0][y0]) {	
	if(MaxNumMergeCand > 1)	
	merge_idx[x0][y0]	ae(v)
	} else {	
	.....	
	}	
	}	
	}	

[도16]


1601
제1 스테이지
1602
제2 스테이지
1603
제3 스테이지
start
Cu_skip_flag=1
Cu_skip_flag=0
merge_idx
merge_flag
merge_flag = 1
merge_flag = 0
merge_idx
inter_pred_idc
inter_pred_idc !=PRED_L1
inter_pred_idc =PRED_L1
inter_pred_idc =PRED_BI
ref_idx_l0 mvd_coding (abs_mvd_greater0_flag, abs_mvd_greater1_flag, abs_mvd_minus2, mvd_sign_flag) mvp_l0_flag
ref_idx_l1 mvd_coding (abs_mvd_greater0_flag, abs_mvd_greater1_flag, abs_mvd_minus2, mvd_sign_flag) mvp_l1_flag
ref_idx_l1 mvd_coding (abs_mvd_greater0_flag, abs_mvd_greater1_flag, abs_mvd_minus2, mvd_sign_flag) mvp_l1_flag

[도17a]

prediction_unit(x0, y0, nPbW, nPbH) {	Descriptor
if(cu_skip_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
(LA) merge_idx[x0][y0](LR)	ae(v)
} else { /* MODE_INTER */	
if(PartMode != PART_2Nx2N)	
(LA) merge_flag[x0][y0](LR)	ae(v)
if(merge_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
(LA) merge_idx[x0][y0](LR)	ae(v)
} else {	
if(slice_type == B)	
(LA) inter_pred_idc[x0][y0](LR)	ae(v)
if(inter_pred_idc[x0][y0] != PRED_L1) {	
if(num_ref_idx_l0_active_minus1 > 0)	
(LA) ref_idx_l0[x0][y0] (LR)	
(LA)	
abs_mvd_greater0_l0_flag[0]	ae(v)
abs_mvd_greater0_l0_flag[1]	ae(v)
(LR)	
(LA)	
if(abs_mvd_greater0_l0_flag[0])	
abs_mvd_greater1_l0_flag[0]	ae(v)
if(abs_mvd_greater0_l0_flag[1])	
abs_mvd_greater1_l0_flag[1]	ae(v)
(LR)	

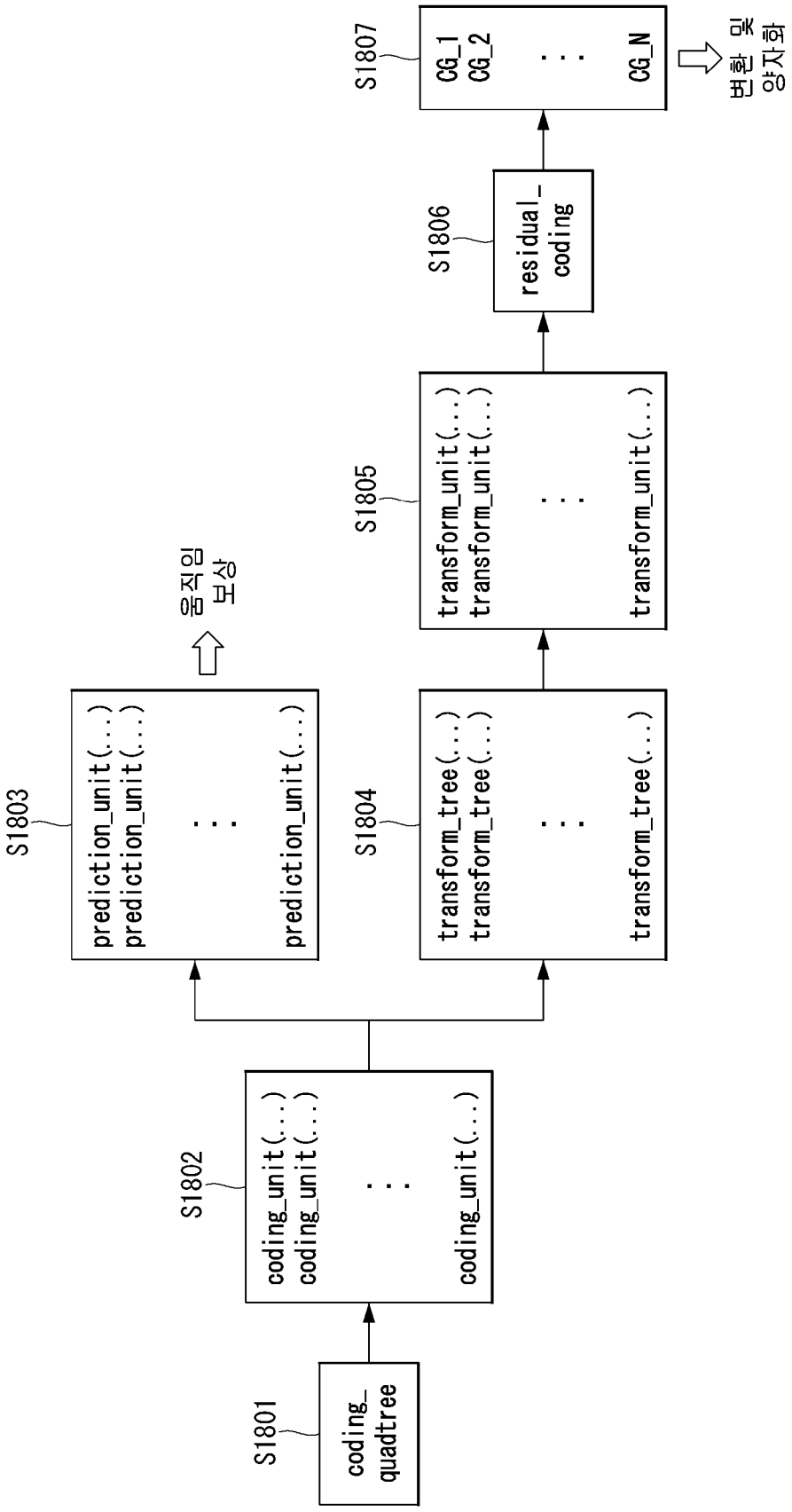
[도17b]

if(abs_mvd_greater0_l0_flag[0]) {	
if(abs_mvd_greater1_l0_flag[0])	
abs_mvd_minus2_l0[0]	ae(v)
mvd_sign_l0_flag[0]	ae(v)
}	
if(abs_mvd_greater0_l0_flag[1]) {	
if(abs_mvd_greater1_l0_flag[1])	
abs_mvd_minus2_l0[1]	ae(v)
mvd_sign_l0_flag[1]	ae(v)
}	
(LA) mvp_l0_flag[x0][y0] (LR)	ae(v)
}	
if(inter_pred_idc[x0][y0] != PRED_L0) {	
if(num_ref_idx_l1_active_minus1 > 0)	
(LA) ref_idx_l1[x0][y0] (LR)	ae(v)
if(mvd_l1_zero_flag && inter_pred_idc[x0][y0] == PRED_BI) {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
} else {	
(LA)	
abs_mvd_greater0_l1_flag[0]	ae(v)
abs_mvd_greater0_l1_flag[1]	ae(v)
(LR)	

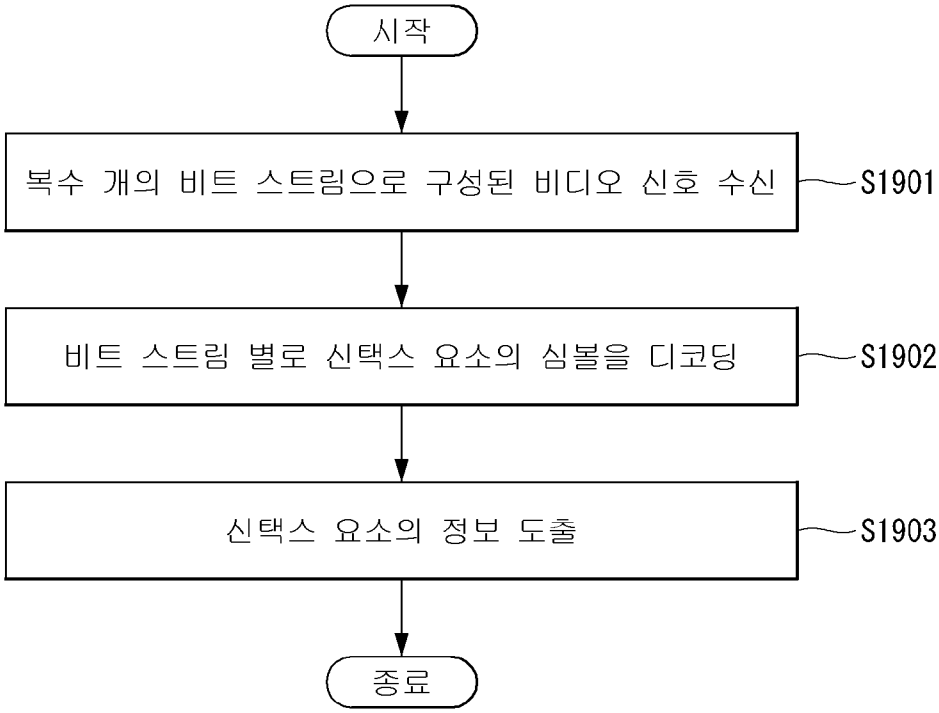
[도17c]

(LA)	
if(abs_mvd_greater0_l1_flag[0])	
abs_mvd_greater1_l1_flag[0]	ae(v)
if(abs_mvd_greater0_l1_flag[1])	
abs_mvd_greater1_l1_flag[1]	ae(v)
(LR)	
if(abs_mvd_greater0_l1_flag[0]) {	
if(abs_mvd_greater1_l1_flag[0])	
abs_mvd_minus2_l1[0]	ae(v)
mvd_sign_l1_flag[0]	ae(v)
}	
if(abs_mvd_greater0_l1_flag[1]) {	
if(abs_mvd_greater1_l1_flag[1])	
abs_mvd_minus2_l1[1]	ae(v)
mvd_sign_l1_flag[1]	ae(v)
}	
}	
(LA) mvp_l1_flag[x0][y0] (LR)	ae(v)
}	
}	
}	
}	

[도18]


S1801
coding_quadtree
S1802
coding_unit(...)
coding_unit(...)
coding_unit(...)
S1803
prediction_unit(...)
prediction_unit(...)
prediction_unit(...)
움직임 보상
S1804
transform_tree(...)
transform_tree(...)
transform_tree(...)
S1805
transform_unit(...)
transform_unit(...)
transform_unit(...)
S1806
residual_coding
S1807
CG_1
CG_2
CG_N
변환 및 양자화

[도19]


시작
복수 개의 비트 스트림으로 구성된 비디오 신호 수신
S1901
비트 스트림 별로 신택스 요소의 심볼을 디코딩
S1902
신택스 요소의 정보 도출
S1903
종료

INTERNATIONAL SEARCH REPORT

International application No.
PCT/KR2017/004818

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/13(2014.01)i, H04N 19/70(2014.01)i, H04N 19/176(2014.01)i, H04N 19/68(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 19/13; H03M 7/30; H04N 19/174; H04N 19/436; H04N 19/70; H04N 19/124; G06T 9/00; H04N 19/30; H04N 19/176; H04N 19/68

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: entropy, encoding, parallel, syntax, bit stream

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	KR 10-2013-0096497 A (IUCF-HYU (INDUSTRY-UNIVERSITY COOPERATION FOUNDATION HANYANG UNIVERSITY)) 30 August 2013 See paragraphs [0031]-[0032], [0037]-[0076]; claims 1, 4; and figures 6-13.	1-18
A	KR 10-2012-0004319 A (ELECTRONICS AND TELECOMMUNICATIONS RESEARCH INSTITUTE et al.) 12 January 2012 See paragraphs [0028]-[0031]; and figures 12-13.	1-18
A	KR 10-1726274 B1 (ELECTRONICS AND TELECOMMUNICATIONS RESEARCH INSTITUTE et al.) 18 April 2017 See paragraphs [0008], [0070]-[0135]; claim 1; and figures 4-7.	1-18
A	KR 10-2016-0070768 A (QUALCOMM INCORPORATED) 20 June 2016 See paragraphs [0026], [0054]-[0125], [0167]; claim 15; and figures 1-5.	1-18
A	KR 10-1718488 B1 (GE VIDEO COMPRESSION, LLC.) 21 March 2017 See paragraphs [0034]-[0077]; and figures 3-10.	1-18

☐ 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
13 FEBRUARY 2018 (13.02.2018)	**02 MARCH 2018 (02.03.2018)**

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.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/KR2017/004818

Patent document cited in search report	Publication date	Patent family member	Publication date
KR 10-2013-0096497 A	30/08/2013	KR 10-1485219 B1	29/01/2015
KR 10-2012-0004319 A	12/01/2012	NONE	
KR 10-1726274 B1	18/04/2017	KR 10-1726276 B1	17/04/2017
		KR 10-1797055 B1	16/11/2017
		KR 10-2015-0099489 A	31/08/2015
		KR 10-2017-0021814 A	28/02/2017
KR 10-2016-0070768 A	20/06/2016	CN 105637864 A	01/06/2016
		EP 3058729 A1	24/08/2016
		JP 2016-533696 A	27/10/2016
		MX 2016004644 A	05/08/2016
		TW 201528788 A	16/07/2015
		US 2015-0103920 A1	16/04/2015
		WO 2015-057677 A1	23/04/2015
KR 10-1718488 B1	21/03/2017	AU 2013-211002 A1	18/09/2014
		AU 2013-211002 B2	14/01/2016
		AU 2016-202208 A1	05/05/2016
		AU 2016-202208 B2	04/05/2017
		AU 2017-210565 A1	24/08/2017
		CA 2861951 A1	25/07/2013
		CL 2014001886 A1	21/11/2014
		CN 104081781 A	01/10/2014
		EP 2805491 A2	26/11/2014
		HK 1204182 A1	06/11/2015
		IL 233699 A	30/04/2017
		IL 251266 A	29/05/2017
		IN 1699KON2014 A	23/10/2015
		JP 2015-507899 A	12/03/2015
		JP 2016-158267 A	01/09/2016
		KR 10-1752879 B1	30/06/2017
		KR 10-2015-0003407 A	08/01/2015
		KR 10-2017-0077295 A	05/07/2017
		KR 10-2017-0078850 A	07/07/2017
		KR 10-2017-0078851 A	07/07/2017
		KR 10-2017-0078852 A	07/07/2017
		MX 2014008695 A	21/11/2014
		PH 12014501660 A1	13/10/2014
		RU 2014134041 A	20/03/2016
		RU 2610291 C2	08/02/2017
		SG 10201606621 A	29/09/2016
		SG 11201404251 A	28/08/2014
		TW 201336314 A	01/09/2013
		TW 201728174 A	01/08/2017
		TW I559744 B	21/11/2016
		UA 114618 C2	10/07/2017
		US 2014-0334557 A1	13/11/2014

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/KR2017/004818

Patent document cited in search report	Publication date	Patent family member	Publication date
		WO 2013-107906 A2	25/07/2013
		WO 2013-107906 A3	26/09/2013

국 제 조 사 보 고 서

국제출원번호
PCT/KR2017/004818

A. 발명이 속하는 기술분류(국제특허분류(IPC))

H04N 19/13(2014.01)i, H04N 19/70(2014.01)i, H04N 19/176(2014.01)i, H04N 19/68(2014.01)i

B. 조사된 분야

조사된 최소문헌(국제특허분류를 기재)
H04N 19/13; H03M 7/30; H04N 19/174; H04N 19/436; H04N 19/70; H04N 19/124; G06T 9/00; H04N 19/30; H04N 19/176; H04N 19/68

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: 엔트로피, 부호화, 병렬, 신택스, 비트스트림

C. 관련 문헌

카테고리*	인용문헌명 및 관련 구절(해당하는 경우)의 기재	관련 청구항
X	KR 10-2013-0096497 A (한양대학교 산학협력단) 2013.08.30 단락 [0031]-[0032], [0037]-[0076]; 청구항 1, 4; 및 도면 6-13 참조.	1-18
A	KR 10-2012-0004319 A (한국전자통신연구원 등) 2012.01.12 단락 [0028]-[0031]; 및 도면 12-13 참조.	1-18
A	KR 10-1726274 B1 (한국전자통신연구원 등) 2017.04.18 단락 [0008], [0070]-[0135]; 청구항 1; 및 도면 4-7 참조.	1-18
A	KR 10-2016-0070768 A (퀄컴 인코포레이티드) 2016.06.20 단락 [0026], [0054]-[0125], [0167]; 청구항 15; 및 도면 1-5 참조.	1-18
A	KR 10-1718488 B1 (지이 비디오 컴프레션, 엘엘씨) 2017.03.21 단락 [0034]-[0077]; 및 도면 3-10 참조.	1-18

☐ 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

"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
2018년 02월 13일 (13.02.2018)	2018년 03월 02일 (02.03.2018)

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

심사관
안정환
전화번호 +82-42-481-8633

국제조사보고서
대응특허에 관한 정보

국제출원번호
PCT/KR2017/004818

국제조사보고서에서 인용된 특허문헌	공개일	대응특허문헌	공개일
KR 10-2013-0096497 A	2013/08/30	KR 10-1485219 B1	2015/01/29
KR 10-2012-0004319 A	2012/01/12	없음	
KR 10-1726274 B1	2017/04/18	KR 10-1726276 B1	2017/04/17
		KR 10-1797055 B1	2017/11/16
		KR 10-2015-0099489 A	2015/08/31
		KR 10-2017-0021814 A	2017/02/28
KR 10-2016-0070768 A	2016/06/20	CN 105637864 A	2016/06/01
		EP 3058729 A1	2016/08/24
		JP 2016-533696 A	2016/10/27
		MX 2016004644 A	2016/08/05
		TW 201528788 A	2015/07/16
		US 2015-0103920 A1	2015/04/16
		WO 2015-057677 A1	2015/04/23
KR 10-1718488 B1	2017/03/21	AU 2013-211002 A1	2014/09/18
		AU 2013-211002 B2	2016/01/14
		AU 2016-202208 A1	2016/05/05
		AU 2016-202208 B2	2017/05/04
		AU 2017-210565 A1	2017/08/24
		CA 2861951 A1	2013/07/25
		CL 2014001886 A1	2014/11/21
		CN 104081781 A	2014/10/01
		EP 2805491 A2	2014/11/26
		HK 1204182 A1	2015/11/06
		IL 233699 A	2017/04/30
		IL 251266 A	2017/05/29
		IN 1699KON2014 A	2015/10/23
		JP 2015-507899 A	2015/03/12
		JP 2016-158267 A	2016/09/01
		KR 10-1752879 B1	2017/06/30
		KR 10-2015-0003407 A	2015/01/08
		KR 10-2017-0077295 A	2017/07/05
		KR 10-2017-0078850 A	2017/07/07
		KR 10-2017-0078851 A	2017/07/07
		KR 10-2017-0078852 A	2017/07/07
		MX 2014008695 A	2014/11/21
		PH 12014501660 A1	2014/10/13
		RU 2014134041 A	2016/03/20
		RU 2610291 C2	2017/02/08
		SG 10201606621 A	2016/09/29
		SG 11201404251 A	2014/08/28
		TW 201336314 A	2013/09/01
		TW 201728174 A	2017/08/01
		TW I559744 B	2016/11/21
		UA 114618 C2	2017/07/10
		US 2014-0334557 A1	2014/11/13

국제조사보고서
대응특허에 관한 정보

국제출원번호
PCT/KR2017/004818

국제조사보고서에서 인용된 특허문헌	공개일	대응특허문헌	공개일
		WO 2013-107906 A2	2013/07/25
		WO 2013-107906 A3	2013/09/26