(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2017-0140432
(43) 공개일자 2017년12월20일

(51) 국제특허분류(Int. Cl.)
H04N 19/70 (2014.01) *H04N 19/13* (2014.01)
H04N 19/169 (2014.01) *H04N 19/174* (2014.01)
H04N 19/436 (2014.01) *H04N 19/46* (2014.01)
H04N 19/463 (2014.01) *H04N 19/593* (2014.01)
H04N 19/91 (2014.01)
(52) CPC특허분류
H04N 19/70 (2015.01)
H04N 19/13 (2015.01)
(21) 출원번호 10-2017-7035615(분할)
(22) 출원일자(국제) 2013년04월15일
심사청구일자 없음
(62) 원출원 특허 10-2016-7034316
원출원일자(국제) 2013년04월15일
심사청구일자 2016년12월07일
(85) 번역문제출일자 2017년12월11일
(86) 국제출원번호 PCT/EP2013/057798
(87) 국제공개번호 WO 2013/153226
국제공개일자 2013년10월17일
(30) 우선권주장
61/624,098 2012년04월13일 미국(US)
61/666,185 2012년06월29일 미국(US)

(71) 출원인
지이 비디오 컴프레션, 엘엘씨
미국 뉴욕 12211 올버니 사우스우즈 블러바드 8
(72) 발명자
쉬를, 토마스
독일 10437 베를린 둔커슈트라쎄 72
게오르게, 발레리
독일 10365 베를린 존-지크-슈트라쎄 24
(뒷면에 계속)
(74) 대리인
윤의섭, 김수진

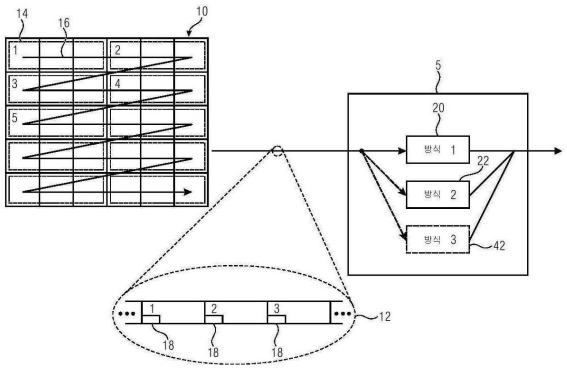
전체 청구항 수 : 총 26 항

(54) 발명의 명칭 **저지연 화상 코딩**

(57) 요 약

엔트로피 디코딩이 관련되는 한 각각의 슬라이스 외부의 화상의 영역과 관계없는 어떠한 슬라이스들이 완전히 코딩되거나/디코딩되는지에 따라 보통의 슬라이스 개념이 서로 다른 방식의 슬라이스들, 즉 슬라이스 경계들을 가로질러 상호의존성을 허용하는 의존성 슬라이스들로 불리는 슬라이스들, 및 예를 들면, 정상적인 슬라이스들로 불리는 그렇지 않은 나머지들을 위하여 포기되면 감소된 종단간 지연으로 파면 병렬 처리와 같은 병렬 처리 개념들이 실현된다. 상기 관점(aspect)과 결합되거나 또는 그렇지 않거나, 파면 병렬 처리 개념은 파면 병렬 처리 엔트리 지점들을 위치시키기 위하여 슬라이스들의 출발 구문 부를 사용함으로써 더 효율적이 된다.

대 표 도



(52) CPC특허분류
H04N 19/174 (2015.01)
H04N 19/188 (2015.01)
H04N 19/436 (2015.01)
H04N 19/46 (2015.01)
H04N 19/463 (2015.01)
H04N 19/593 (2015.01)
H04N 19/91 (2015.01)
(72) 발명자
헨켈, 아나스타샤
독일 10245 베를린 파렌하이트슈트라쎄 8
마르페, 데트레브
독일 12161 베를린 쉬드베스트코르소 70
그륀버그, 카르스텐
독일 13599 베를린 아딕케스슈트라쎄 43
스쿠핀, 로버트
독일 10555 베를린 슐레스비히 우퍼 5

명 세 서

청구범위

청구항 1

화상(10)이 분할되는 슬라이스들(14)의 유닛들 내에 상기 화상(10)이 코딩되는 데이터스트림(12)으로부터 상기 화상(10)을 재구성하기 위한 디코더에 있어서,

상기 디코더는 슬라이스 순서(16)에 따라 상기 데이터스트림(12)으로부터 상기 슬라이스들(14)을 디코딩하도록 구성되고,

상기 디코더는:

적어도 두 가지 방식(20, 22) 중 어느 하나에 따라 현재 슬라이스를 디코딩하기 위하여 상기 슬라이스들의 상기 현재 슬라이스 내의 구문 요소 부(18)에 응답하며,

상기 적어도 두 가지 방식 중 첫 번째 방식(20)에 따라, 슬라이스 경계들을 가로지른 콘텍스트들의 도출, 상기 콘텍스트들의 심볼 확률들의 연속적인 업데이트들 및 이전에 디코딩된 슬라이스의 심볼 확률들의 저장된 상태에 따른 상기 심볼 확률들의 초기화(38, 40)를 포함하는 콘텍스트 적응적 엔트로피 디코딩(24), 및 상기 슬라이스 경계들을 가로지른 예측 디코딩을 사용하여 상기 데이터스트림(12)으로부터 상기 현재 슬라이스를 디코딩하며,

상기 적어도 두 가지 방식 중 두 번째 방식(22)에 따라, 상기 슬라이스 경계들을 가로지르지 않도록 하기 위한 콘텍스트들의 도출의 제한, 상기 콘텍스트들의 심볼 확률들의 연속적인 엡데이트 및 어떠한 이전에 디코딩된 슬라이스와 관계없는 심볼 확률들의 초기화를 갖는 콘텍스트 적응적 엔트로피 디코딩, 및 슬라이스 경계들을 가로지르지 않도록 하기 위한 예측 디코딩의 제한을 갖는 예측 디코딩을 사용하여 상기 데이터스트림(12)으로부터 상기 현재 슬라이스를 디코딩하도록 구성되는 것을 특징으로 하는 디코더.

청구항 2

제 1항에 있어서, 상기 화상(10)은 상기 화상(10)은 열들과 행들로 배치되고 서로 중에서 정의되는 래스터 스캔 순서(36)를 갖는 코딩 블록들로 분할되고, 상기 디코더는 서브셋들이 슬라이스 순서에 따른 래스터 스캔 순서(36)를 따라 서로 뒤따르도록 하기 위하여 각각의 슬라이스(14)를 상기 래스터 스캔 순서(36) 내의 코딩 블록들(32)의 연속적인 서브셋과 연관시키도록 구성되는 것을 특징으로 하는 디코더.

청구항 3

제 2항에 있어서, 상기 디코더는 상기 래스터 스캔 순서(36)에 따라 열 내의 두 번째 코딩 블록(32)까지 상기 이전에 디코딩된 슬라이스의 콘텍스트 적응적 엔트로피 디코딩에서 획득되는 것과 같은 심볼 확률들을 저장하고, 상기 첫 번째 방식에 따라 상기 현재 슬라이스의 상기 콘텍스트 적응적 엔트로피 디코딩을 위한 상기 심볼 확률들의 초기화에서, 상기 현재 슬라이스와 관련된 코딩 블록들(32)의 상기 연속적인 서브셋들의 첫 번째 코딩 블록이 상기 래스터 스캔 순서에 따른 열의 첫 번째 코딩 블록(32)에 대하여 검사하도록 구성될 수 있으며, 만일 그렇다면, 상기 래스터 스캔 순서(36)에 따라 열의 두 번째 코딩 블록까지 상기 이전에 디코딩된 슬라이스의 상기 콘텍스트 적응적 엔트로피 디코딩에서 획득되는 것과 같은 상기 저장된 심볼 확률들에 따라 상기 현재 슬라이스의 상기 콘텍스트 적응적 엔트로피 디코딩을 위한 상기 심볼 확률들을 초기화하고, 만일 그렇지 않으면, 상기 이전에 디코딩된 슬라이스의 끝까지 상기 이전에 디코딩된 슬라이스의 상기 콘텍스트 적응적 엔트로피 디코딩에서 획득되는 것과 같은 심볼 확률들에 따라 상기 현재 슬라이스의 상기 콘텍스트 적응적 엔트로피 디코딩을 위한 상기 심볼 확률들을 초기화하도록 구성되는 것을 특징으로 하는 디코더.

청구항 4

제 1항 내지 3항 중 어느 한 항에 있어서, 상기 디코더는 적어도 세 가지 방식 중 어느 하나에 따라, 즉 첫 번

째(20) 및 세 번째 방식(42) 또는 두 번째 방식(22) 중 어느 하나에서 상기 현재 슬라이스를 디코딩하기 위하여 상기 슬라이스들(14)의 상기 현재 슬라이스 내의 상기 구문 요소 부(18)에 응답하도록 구성되고,

상기 디코더는 상기 세 번째 방식(42)에 따라, 상기 슬라이스 경계들을 가로지르지 않도록 하기 위한 상기 콘텍스트들의 도출의 제한, 상기 콘텍스트들의 심볼 확률들의 연속적인 엡데이트 및 어떠한 이전에 디코딩된 슬라이스와 관계없는 상기 심볼 확률들의 초기화를 갖는 콘텍스트 적응적 엔트로피 디코딩, 및 상기 슬라이스 경계들을 가로지른 예측 디코딩을 사용하여 상기 데이터스트림으로부터 상기 현재 슬라이스를 디코딩하도록 구성되며,

상기 첫 번째 및 세 번째 방식 중 어느 하나가 구문 요소에 따라 선택되는 것을 특징으로 하는 디코더.

청구항 5

제 1항 내지 3항 중 어느 한 항에 있어서, 상기 디코더는 첫 번째 제네릭 작동 방식에 따른 각각의 슬라이스를 위한 상기 구문 요소에 대한 응답성의 실행, 및 두 번째 제네릭 작동 방식에 따른 상기 첫 번째 방식 이외의 적어도 두 가지 방식 중 다른 하나의 필연적인 사용을 갖는, 적어도 두 가지 제네릭 작동 방식 중 중 어느 하나에서 작동하도록 하기 위하여 제네릭 구문 요소에 응답하도록 구성되는 것을 특징으로 하는 디코더.

청구항 6

제 2항에 있어서, 상기 디코더는 상기 첫 번째 및 두 번째 방식에 따라 필연적으로 그리고 중단됨이 없이, 현재 슬라이스의 시작부터 끝까지 심볼 확률들을 연속적으로 계속해서 업데이트하도록 구성되는 것을 특징으로 하는 디코더.

청구항 7

제 2항에 있어서, 상기 디코더는 상기 이전에 인코딩된 슬라이스의 끝까지 상기 이전에 인코딩된 슬라이스의 콘텍스트 적응적 엔트로피 인코딩에서 획득되는 것과 같은 심볼 확률들을 저장하고, 상기 첫 번째 방식에 따른 상기 현재 슬라이스의 상기 콘텍스트 적응적 엔트로피 인코딩을 위한 상기 심볼 확률들의 초기화에 있어서, 저장된 심볼 확률들에 따라 상기 현재 슬라이스의 상기 콘텍스트 적응적 엔트로피 인코딩을 위한 상기 심볼 확률들을 초기화하도록 구성되는 것을 특징으로 하는 디코더.

청구항 8

제 1항 내지 7항 중 어느 한 항에 있어서, 상기 디코더는 상기 첫 번째 및 두 번째 방식에서, 상기 화상이 세분화되는 타일들 내로의 예측 인코딩을 제한하도록 구성되는 것을 특징으로 하는 디코더.

청구항 9

제 1항 내지 8항 중 어느 한 항에 있어서, 상기 디코더는 상기 첫 번째 및 두 번째 방식에서, 상기 현재 슬라이스의 병렬 서브섹션들로의 세분화를 나타내는 현재 슬라이스로부터의 정보를 판독하고, 상기 첫 번째 병렬 서브섹션의 끝에서 상기 콘텍스트 적응적 엔트로피 디코딩을 중단하며, 상기 첫 번째 방식에서, 상기 선행 병렬 서브섹션의 심볼 확률들의 저장된 상태에 따른 상기 심볼 확률들의 초기화 및, 상기 두 번째 방식에서, 어떠한 이전에 디코딩된 슬라이스 및 어떠한 이전에 디코딩된 병렬 서브섹션과 관계없는 상기 심볼 확률들의 초기화를 포함하는 어떠한 뒤따르는 병렬 서브섹션의 시작에서 상기 콘텍스트 적응적 엔트로피 디코딩을 새로 재개하도록 구성되는 것을 특징으로 하는 디코더.

청구항 10

제 1항 내지 9항 중 어느 한 항에 있어서, 상기 디코더는 상기 적어도 두 가지 방식 중 첫 번째 방식(20)에 따

라, 상기 현재 슬라이스를 위하여 상기 두 번째 방식에서 디코딩된 선행 슬라이스로부터 슬라이스 헤더 구문의 부를 복사하도록 구성되는 것을 특징으로 하는 디코더.

청구항 11

선행 항 중 어느 한 항에 있어서, 상기 디코더는 파면 병렬 처리 과정을 사용하여 데이터스트림(12)으로부터 상기 화상(10)을 재구성하도록 구성되고, 각각의 슬라이스(14)는 상기 화상(10) 내의 상기 각각의 슬라이스의 디코딩 시작의 위치를 나타내는 출발 구문 부(400)를 포함하며,

상기 디코더는 상기 슬라이스의 출발 구문 부들을 사용하여 상기 화상의 왼쪽 면에서 시작하는 슬라이스들의 식별에 의해, 상기 슬라이스들이 그룹핑되는 파면 병렬 처리 서브스트림들의 엔트리 지점들을 식별하고, 상기 슬라이스 순서에 따라 상기 파면 병렬 처리 서브스트림들의 디코딩을 순차적으로 시작하는 스태그드 방식으로 상기 파면 병렬 처리 서브스트림들을 병렬 디코딩하도록 구성되는 것을 특징으로 하는 디코더.

청구항 12

화상(10)이 분할되는 슬라이스들(14)의 유닛들 내에 상기 화상(10)이 코딩되는 데이터스트림(12) 내로 상기 화상(10)을 인코딩하기 위한 인코더에 있어서, 상기 인코더는 슬라이스 순서(16)에 따라 상기 데이터스트림(12)으로부터 상기 슬라이스들(14)을 인코딩하도록 구성되고,

상기 인코더는;

구문 요소 부가 적어도 두 가지 방식(20, 22) 중 어느 하나에 따라 코딩되는 현재 슬라이스를 시그널링하도록 하기 위하여 상기 슬라이스들의 상기 현재 슬라이스를 위한 상기 구문 요소 부(18)를 결정하고, 이를 상기 슬라이스들의 현재 슬라이스 내로 코딩하고,

만일 상기 현재 슬라이스가 상기 적어도 두 가지 방식 중 첫 번째 방식(20)에 따라 코딩되면, 슬라이스 경계들을 가로지르는 콘텍스트들의 도출, 상기 콘텍스트들의 심볼 확률들의 연속적인 업데이트 및 이전에 인코딩된 슬라이스의 심볼 확률들의 저장된 상태에 따라 상기 심볼 확률들의 초기화(38, 40)를 포함하는 콘텍스트 적응적 엔트로피 인코딩(24), 및 슬라이스 경계들을 가로지르는 예측 인코딩을 사용하여 상기 현재 슬라이스를 상기 데이터스트림(12) 내로 인코딩하며,

만일 상기 현재 슬라이스가 상기 적어도 두 가지 방식 중 두 번째 방식(22)에 따라 코딩되면, 상기 슬라이스 경계들을 가로지르기 않도록 하기 위한 콘텍스트들의 도출의 제한, 상기 콘텍스트들의 심볼 확률들의 연속적인 업데이트 및 어떠한 이전에 인코딩된 슬라이스와 관계없는 상기 심볼 확률들의 초기화(38, 40)를 갖는 콘텍스트 적응적 엔트로피 인코딩(24), 및 슬라이스 경계들을 가로지르지 않도록 하기 위한 예측 인코딩의 제한을 갖는 예측 인코딩을 사용하여 상기 현재 슬라이스를 상기 데이터스트림(12) 내로 인코딩하도록 구성되는 것을 특징으로 하는 인코더.

청구항 13

제 12항에 있어서, 상기 화상(10)은 상기 화상(10)은 열들과 행들로 배치되고 서로 중에서 정의되는 래스터 스캔 순서(36)를 갖는 코딩 블록들로 분할되고, 상기 인코더는 서브셋들이 슬라이스 순서에 따른 래스터 스캔 순서(36)를 따라 서로 뒤따르도록 하기 위하여 각각의 슬라이스(14)를 상기 래스터 스캔 순서(36) 내의 코딩 블록들(32)의 연속적인 서브셋과 연관시키도록 구성되는 것을 특징으로 하는 인코더.

청구항 14

제 13항에 있어서, 상기 인코더는 상기 래스터 스캔 순서(36)에 따라 열 내의 두 번째 코딩 블록(32)까지 상기 이전에 인코딩된 슬라이스의 콘텍스트 적응적 엔트로피 인코딩에서 획득되는 것과 같은 심볼 확률들을 저장하고, 상기 첫 번째 방식에 따라 상기 현재 슬라이스의 상기 콘텍스트 적응적 엔트로피 인코딩을 위한 상기 심볼 확률들의 초기화에서, 상기 현재 슬라이스와 관련된 코딩 블록들(32)의 상기 연속적인 서브셋들의 첫 번째

코딩 블록이 상기 래스터 스캔 순서에 따른 열의 첫 번째 코딩 블록(32)에 대하여 검사하도록 구성될 수 있으며, 만일 그렇다면, 상기 래스터 스캔 순서(36)에 따라 열의 두 번째 코딩 블록까지 상기 이전에 인코딩된 슬라이스의 상기 콘텍스트 적응적 엔트로피 인코딩에서 획득되는 것과 같은 상기 저장된 심볼 확률들에 따라 상기 현재 슬라이스의 상기 콘텍스트 적응적 엔트로피 인코딩을 위한 상기 심볼 확률들을 초기화하고, 만일 그렇지 않으면, 상기 이전에 인코딩된 슬라이스의 끝까지 상기 이전에 인코딩된 슬라이스의 상기 콘텍스트 적응적 엔트로피 인코딩에서 획득되는 것과 같은 심볼 확률들에 따라 상기 현재 슬라이스의 상기 콘텍스트 적응적 엔트로피 인코딩을 위한 상기 심볼 확률들을 초기화하도록 구성되는 것을 특징으로 하는 인코더.

청구항 15

제 12항 내지 14항 중 어느 한 항에 있어서, 상기 인코더는 상기 현재 슬라이스가 적어도 세 가지 방식 중 하나에 따라, 즉, 첫 번째(20) 및 세 번째 방식(42) 또는 두 번째 방식(20)으로 그것들 내로 코딩되도록 시그널링되기 위하여 상기 슬라이스들(14)의 상기 현재 슬라이스 내로 상기 구문 요소 부(18)를 코딩하고,

상기 인코더는 상기 세 번째 방식(42)에 따라, 상기 슬라이스 경계들을 가로지르지 않도록 하기 위한 콘텍스트의 도출의 제한, 상기 콘텍스트들의 심볼 확률들의 연속적인 업데이트 및 어떠한 이전에 인코딩된 슬라이스와 관계없는 상기 심볼 확률들의 초기화를 갖는 콘텍스트 적응적 엔트로피 인코딩, 및 상기 슬라이스 경계들을 가로지르는 예측 코딩을 사용하여 상기 현재 슬라이스를 상기 데이터스트림 내로 인코딩하도록 구성되며,

상기 인코더는 구문 요소를 사용하여 상기 첫 번째 및 세 번째 방식 중 어느 하나 사이를 구별하는 것을 특징으로 하는 인코더.

청구항 16

제 12항 내지 15항 중 어느 한 항에 있어서, 성기 인코더는 제네릭 구문 요소를 결정하고, 상기 제네릭 구문 요소에 따른 적어도 두 가지 제네릭 작동 방식, 즉, 각각의 슬라이스를 위하여 상기 구문 요소 부의 코딩을 실행하는, 첫 번째 제네릭 작동 방식, 및 상기 첫 번째 방식 이외의 상기 적어도 두 가지 방식 중 다른 하나를 필연적으로 사용하는, 두 번째 제네릭 작동 방식 중 어느 하나에서의 작동으로, 상기 데이터스트림 내로 이를 기록하도록 구성되는 것을 특징으로 하는 인코더.

청구항 17

제 13항에 있어서, 상기 인코더는 상기 첫 번째 및 두 번째 방식에 따라 필연적으로 또는 중단됨이 없이, 상기 현재 슬라이스의 시작부터 끝까지 상기 심볼 확률들을 연속적으로 계속해서 업데이트하도록 구성되는 것을 특징으로 하는 인코더.

청구항 18

제 13항에 있어서, 상기 인코더는 상기 이전에 인코딩된 슬라이스의 끝까지 상기 이전에 인코딩된 슬라이스의 콘텍스트 적응적 엔트로피 인코딩에서 획득되는 것과 같은 심볼 확률들을 저장하고, 상기 첫 번째 방식에 따른 상기 현재 슬라이스의 상기 콘텍스트 적응적 엔트로피 인코딩을 위한 상기 심볼 확률들을 초기화하는데 있어서, 저장된 심볼 확률들에 따라 상기 현재 슬라이스의 사익 콘텍스트 적응적 엔트로피 인코딩을 위한 상기 심볼 확률들을 초기화하도록 구성되는 것을 특징으로 하는 인코더.

청구항 19

제 12항 내지 18항 중 어느 한 항에 있어서, 상기 인코더는 상기 첫 번째 및 두 번째 방식에서, 상기 화상이 세분화되는 타일들 내로의 예측 인코딩을 제한하도록 구성되는 것을 특징으로 하는 인코더.

청구항 20

파면 병렬 처리를 사용하여, 화상(10)이 분할되는 슬라이스들(14)의 유닛들 내에 상기 화상(10)이 코딩되는 데이터스트림(12)으로부터 상기 화상(10)을 재구성하기 위한 디코더에 있어서, 상기 디코더는 슬라이스 순서(16)에 따라 상기 데이터스트림(12)으로부터 상기 슬라이스들(14)을 디코딩하도록 구성되고, 각각의 상기 슬라이스(14)는 상기 화상(10) 내의 상기 각각의 슬라이스의 디코딩 시작의 위치를 나타내는 출발 구문 부(400)를 포함하며,

상기 디코더는 상기 슬라이스의 출발 구문 부들(400)을 사용하여 상기 화상의 왼쪽 면에서 시작하는 슬라이스들을 식별함으로써, 상기 슬라이스들이 그룹핑되는 파면 병렬 처리 서브스트림들의 엔트리 지점들을 식별하고, 상기 슬라이스 순서에 따라 상기 파면 병렬 처리 서브스트림들의 디코딩을 순차적으로 시작하는 스태그드 방식으로 상기 파면 병렬 처리 서브스트림들을 병렬 디코딩하도록 구성되는 것을 특징으로 하는 디코더.

청구항 21

파면 병렬 처리를 사용하여, 화상(10)이 분할되는 슬라이스들(14)의 유닛들 내에 상기 화상(10)이 코딩되는 데이터스트림(12) 내로 상기 화상(10)을 코딩하기 위한 인코더에 있어서,

상기 인코더는 슬라이스 순서(16)에 따라 상기 슬라이스들(14)을 상기 데이터스트림(12) 내로 인코딩하도록 구성되고, 상기 인코더는 각각의 상기 슬라이스(14)에 상기 화상(10) 내의 상기 각각의 슬라이스의 코딩 시작의 위치를 나타내는 출발 구문 부(400)를 제공하도록 구성되며,

상기 인코더는 각각의 파면 병렬 처리 서브스트림을 위하여, 슬라이스 순서 내의 상기 첫 번째 슬라이스가 상기 화상의 왼쪽 면에서 시작하도록 상기 슬라이스들을 상기 파면 병렬 처리 서브스트림들 내로 그룹핑하고, 상기 슬라이스 순서에 따라 상기 파면 병렬 처리 서브스트림들의 인코딩을 순차적으로 시작하는 스태그드 방식으로 상기 파면 병렬 처리 서브스트림들을 병렬 인코딩하도록 구성되는 것을 특징으로 하는 인코더.

청구항 22

화상(10)이 분할되는 슬라이스들(14)의 유닛들 내에 상기 화상(10)이 코딩되는 데이터스트림(12)으로부터 상기 화상(10)을 재구성하기 위한 방법에 있어서,

슬라이스 순서(16)에 따라 상기 데이터스트림(12)으로부터 상기 슬라이스들(14)을 디코딩하는 단계를 포함하고, 상기 방법은 적어도 두 가지 방식(20, 22) 중 어느 하나에 따라 현재 슬라이스를 디코딩하기 위하여 상기 슬라이스들의 상기 현재 슬라이스 내의 구문 요소 부(18)에 응답하며,

상기 적어도 두 가지 방식 중 첫 번째 방식(20)에 따라, 상기 현재 슬라이스는 슬라이스 경계들을 가로지른 콘텍스트들의 도출, 상기 콘텍스트들의 심볼 확률들의 연속적인 업데이트들 및 이전에 디코딩된 슬라이스의 심볼 확률들의 저장된 상태에 따른 상기 심볼 확률들의 초기화(38, 40)를 포함하는 콘텍스트 적응적 엔트로피 디코딩(24), 및 상기 슬라이스 경계들을 가로지른 예측 디코딩을 사용하여 상기 데이터스트림(12)으로부터 디코딩되며,

상기 적어도 두 가지 방식 중 두 번째 방식(22)에 따라, 상기 현재 슬라이스는 상기 슬라이스 경계들을 가로지르지 않도록 하기 위한 콘텍스트들의 도출의 제한, 상기 콘텍스트들의 심볼 확률들의 연속적인 엡데이트 및 어떠한 이전에 디코딩된 슬라이스와 관계없는 심볼 확률들의 초기화를 갖는 콘텍스트 적응적 엔트로피 디코딩, 및 슬라이스 경계들을 가로지르지 않도록 하기 위한 예측 디코딩의 제한을 갖는 예측 디코딩을 사용하여 상기 데이터스트림(12)으로부터 디코딩되는 것을 특징으로 하는 데이터스트림으로부터 화상을 재구성하기 위한 방법.

청구항 23

화상(10)이 분할되는 슬라이스들(14)의 유닛들 내에 상기 화상(10)이 코딩되는 데이터스트림(12) 내로 상기 화상(10)을 인코딩하기 위한 방법에 있어서,

상기 방법은 슬라이스 순서(16)에 따라 상기 데이터스트림(12)으로부터 상기 슬라이스들(14)을 인코딩하는 단계를 포함하며, 상기 방법은:

구문 요소 부가 적어도 두 가지 방식(20, 22) 중 어느 하나에 따라 코딩되는 현재 슬라이스를 시그널링하도록 하기 위하여 상기 슬라이스들의 상기 현재 슬라이스를 위한 상기 구문 요소 부(18)를 결정하고, 이를 상기 슬라이스들의 현재 슬라이스 내로 코딩하는 단계; 및

만일 상기 현재 슬라이스가 상기 적어도 두 가지 방식 중 첫 번째 방식(20)에 따라 코딩되면, 슬라이스 경계들을 가로지르는 콘텍스트들의 도출, 상기 콘텍스트들의 심볼 확률들의 연속적인 업데이트 및 이전에 인코딩된 슬라이스의 심볼 확률들의 저장된 상태에 따라 상기 심볼 확률들의 초기화(38, 40)를 포함하는 콘텍스트 적응적 엔트로피 인코딩(24), 및 슬라이스 경계들을 가로지르는 예측 인코딩을 사용하여 상기 현재 슬라이스를 상기 데이터스트림(12) 내로 인코딩하는 단계; 및

만일 상기 현재 슬라이스가 상기 적어도 두 가지 방식 중 두 번째 방식(22)에 따라 코딩되면, 상기 슬라이스 경계들을 가로지르기 않도록 하기 위한 콘텍스트들의 도출의 제한, 상기 콘텍스트들의 심볼 확률들의 연속적인 업데이트 및 어떠한 이전에 인코딩된 슬라이스와 관계없는 상기 심볼 확률들의 초기화(38, 40)를 갖는 콘텍스트 적응적 엔트로피 인코딩(24), 및 슬라이스 경계들을 가로지르지 않도록 하기 위한 예측 인코딩의 제한을 갖는 예측 인코딩을 사용하여 상기 현재 슬라이스를 상기 데이터스트림(12) 내로 인코딩하는 단계;를 더 포함하는 것을 특징으로 하는 데이터스트림 내로 화상을 인코딩하기 위한 방법.

청구항 24

파면 병렬 처리를 사용하여, 화상(10)이 분할되는 슬라이스들(14)의 유닛들 내에 상기 화상(10)이 코딩되는 데이터스트림(12)으로부터 상기 화상(10)을 재구성하기 위한 방법에 있어서,

슬라이스 순서(16)에 따라 상기 데이터스트림(12)으로부터 상기 슬라이스들(14)을 디코딩하는 단계를 포함하되, 각각의 상기 슬라이스(14)는 상기 화상(10) 내의 상기 각각의 슬라이스의 디코딩 시작의 위치를 나타내는 출발 구문 부(400)를 포함하며, 상기 방법은:

상기 슬라이스의 출발 구문 부들(400)을 사용하여 상기 화상의 왼쪽 면에서 시작하는 슬라이스들을 식별함으로써, 상기 슬라이스들이 그룹핑되는 파면 병렬 처리 서브스트림들의 엔트리 지점들을 식별하는 단계; 및

상기 슬라이스 순서에 따라 상기 파면 병렬 처리 서브스트림들의 디코딩을 순차적으로 시작하는 스태그드 방식으로 상기 파면 병렬 처리 서브스트림들을 병렬 디코딩하는 단계;를 더 포함하는 것을 특징으로 하는 데이터스트림으로부터 화상을 재구성하기 위한 방법.

청구항 25

파면 병렬 처리를 사용하여, 화상(10)이 분할되는 슬라이스들(14)의 유닛들 내에 상기 화상(10)이 코딩되는 데이터스트림(12) 내로 상기 화상(10)을 코딩하기 위한 방법에 있어서,

슬라이스 순서(16)에 따라 상기 슬라이스들(14)을 상기 데이터스트림(12) 내로 인코딩하는 단계 및 각각의 상기 슬라이스(14)에 상기 화상(10) 내의 상기 각각의 슬라이스의 코딩 시작의 위치를 나타내는 출발 구문 부(400)를 제공하는 단계를 포함하며,

각각의 파면 병렬 처리 서브스트림을 위하여, 슬라이스 순서 내의 상기 첫 번째 슬라이스가 상기 화상의 왼쪽 면에서 시작하도록 상기 슬라이스들을 상기 파면 병렬 처리 서브스트림들 내로 그룹핑하는 단계; 및

상기 슬라이스 순서에 따라 상기 파면 병렬 처리 서브스트림들의 인코딩을 순차적으로 시작하는 스태그드 방식으로 상기 파면 병렬 처리 서브스트림들을 병렬 인코딩하는 단계;를 더 포함하는 것을 특징으로 하는 데이터스트림 내로 화상을 코딩하기 위한 방법.

청구항 26

컴퓨터 상에서 실행될 때, 제 22항 또는 25항에 따른 방법을 실행하기 위한 프로그램 코드를 갖는 컴퓨터 프로그램.

발명의 설명

기 술 분 야

[0001] 본 발명은 화상들의 저지연 코딩에 관한 것이다.

배 경 기 술

[0003] 현재 고효율 비디오 코딩(High Efficiency Video Coding, HEVC, 이하 HEVC로 표기) 디자인에서, 슬라이스(slice)들, 엔트로피 슬라이스들(이전의 경량 슬라이스들) 타일(tile)들 및 파면 병렬 처리(Wavefront Parallel Processing, WPP)는 병렬화를 위한 툴들로서 포함된다.

[0004] 비디오 인코더들과 디코더들의 병렬화를 위하여 화상-레벨 분할은 다른 접근법들과 비교하여 일부 장점들을 갖는다. H.264/AVC[1] 같은, 이전의 비디오 코더들에서, 화상 파티션들은 코딩 효율과 관련하여 높은 값을 갖는 규칙적인 슬라이스(regular slice)들로만 가능하였다. 스케일러블 병렬 H.264/AVC 디코딩을 위하여 화상 재구성을 위한 매크로블록-레벨 병행성(parallelism) 및 엔트로피 디코딩을 위한 프레임-레벨 병행성을 결합하는 것이 필요하다. 그러나, 이러한 접근법은 화상 지연의 제한된 감소 및 높은 메모리 사용을 제공한다. 이러한 제한들을 극복하기 위하여, 새로운 화상 분할 전략들이 HEVC 코덱 내에 포함되어왔다. 현재의 기준 소프트웨어 버전(HM-6)은 다음의 4가지 서로 다른 접근법을 포함한다: 규칙적이거나 또는 정상적인 슬라이스들, 엔트로피 슬라이스들, 파면 병렬 처리 서브-스트림들 및 타일들. 일반적으로 그러한 화상 파티션들은 HEVC 또는 이들의 서브셋에서 정의되는 것과 같이, 최대 코딩 유닛들(LCU)의 세트, 또는 동의어로, 코딩 트리 유닛(CTU)들의 세트를 포함한다.

[0005] 도 1은 바람직하게는 최대 코딩 유닛들의 열(row, 902) 당 규칙적인 슬라이스(900) 내에 위치되는 화상(898)으로서 도시한다. 규칙적이거나 또는 정상적인 슬라이스들(H.264[1]에서 정의되는 것과 같이)은 최대 코딩 페널티를 갖는데 그 이유는 슬라이스들이 엔트로피 디코딩과 예측 의존성을 깨뜨리기 때문이다.

[0006] 슬라이스들과 같은, 엔트로피 슬라이스들은 엔트로피 디코딩 의존성을 깨뜨리나 슬라이스 경계들을 가로지르기 위한 예측(및 필터링)을 허용한다.

[0007] 파면 병렬 처리에 있어서 화상 파티션들은 이제 삽입되고, 엔트로피 디코딩과 예측 모두 다른 파티션들 내의 블록들로부터 데이터를 사용하도록 허용된다. 이러한 방법으로 코딩 손실이 최소화되며 동시에 파면 병행성이 활용될 수 있다. 그러나, 삽입은 비트스트림 인과율(bitstream causalty)을 침해하는데 그 이유는 이전의 파티션이 그 다음의 디코딩하기 위한 파티션을 필요로 하기 대문이다.

[0008] 도 2는 바람직하게는 수평으로 분할하는 타일들(906)의 두 개의 열(904, 904b)로 세분되는 화상(898)을 도시한다. 타일들은 화상(898)을 타일 행들(912a, b, c)과 열들(904a, b)로 분할하는 수평(908) 및 수직 경계(910)를 정의한다. 규칙적인 슬라이스들(900)과 유사하게, 타일들은 엔트로피 디코딩 및 예측 의존성들을 깨뜨리나, 각각의 타일을 위한 헤더를 요구하지는 않는다.

[0009] 이러한 기술들 각각을 위하여 파티션들의 수는 인코더에 의해 자유롭게 선택될 수 있다. 일반적으로 더 많은 파티션을 갖는 것은 더 높은 압축 손실에 이르게 한다. 그러나 파면 병렬 처리에 있어서 손실 전파는 높지 않으며 따라서 화상 파티션들의 수는 열 당 하나로 고정될 수 있다. 이는 또한 몇몇 장점들에 이르게 한다. 우선, 파면 병렬 처리 비트스트림을 위하여, 인과율이 보장된다. 두 번째로, 디코더 구현들은 해상도에 따라 증가하는, 특정 양의 병행성이 이용가능하다는 것을 추정할 수 있다. 그리고, 마지막으로, 상대적으로 낮은 코딩 손실들을 야기하는, 파면 순서로 디코딩할 때, 콘텍스트 선택과 예측 의존성들 중 어느 것도 깨져서는 안 된다.

[0010] 그러나, 지금까지 변환 개념들에서의 모든 병렬 코딩은 저지연의 유지와 결합하여 고압축 효율의 달성을 제공하는데 실패하였다. 이는 또한 파면 병렬 처리 개념에서도 마찬가지이다. 슬라이스들은 코딩 파이프라인(coding pipeline)에서 전송의 가장 작은 유닛들이며, 일부 파면 병렬 처리 서브스트림들은 여전히 연속적으로 전송되어

야만 한다.

발명의 내용

해결하려는 과제

[0012] 따라서, 종단간(end-to-end) 지연을 더 감소시키거나 또는 사용되는 코딩 오버헤드의 감소에 의해 코딩 효율을 향상시키는 것과 같이, 증가된 효율을 갖는, 예를 들면, 파면 병렬 처리에 따른 병렬 디코딩을 허용하는 화상 코딩 개념을 제공하는 것이 본 발명의 목적이다.

과제의 해결 수단

[0014] 본 발명의 목적은 독립항들의 주제에 의해 달성된다.

[0015] 만일 엔트로피 디코딩이 관련되는 한 각각의 슬라이스 외부의 화상의 영역과 관계없는 어떠한 슬라이스들이 완전히 코딩되거나/디코딩되는지에 따라 보통의 슬라이스 개념이 서로 다른 방식의 슬라이스들, 즉 슬라이스 경계들을 가로질러 상호의존성을 허용하는 의존성 슬라이스들로 불리는 슬라이스들, 및 예를 들면, 정상적인 슬라이스들로 불리는 그렇지 않은 나머지들을 위하여 포기되면 감소된 종단간 지연으로 파면 병렬 처리와 같은 병렬 처리 개념들이 실현될 수 있는 것이 본 발명의 기본 발견이다.

발명의 효과

[0017] 만일 슬라이스들의 출발 구문 부(start syntax portion)들이 파면 병렬 처리 엔트리 지점(entry point)들을 위치시키는데 사용되면 파면 병렬 처리 처리 개념이 더 효율적으로 만들어질 수 있다는 사실이, 첫 번째와 결합되거나 또는 개별적으로 사용될 수 있는 본 발명의 또 다른 기본 발견이다.

도면의 간단한 설명

[0019] 도면들과 관련하여 본 발명의 바람직한 실시 예들이 아래에 설명되는데, 바람직한 실시 예들은 종속항들의 주제이다.

도 1은 바람직하게는 최대 코딩 유닛들의 열 당 규칙적인 슬라이스로 분할되는 화상 또는 화상 내의 매크로블록들을 도시한다.

도 2는 바람직하게는 수평으로 분할되는 타일들의 2열로 분할되는 화상을 도시한다.

도 3은 바람직하게는 슬라이스 또는 네트워크 전송 세그먼트로의 병렬 인코딩된 파티션들의 지정을 도시한다.

도 4는 최소 종단간 지연을 위한 타일 코딩 접근법으로의 프레임의 일반 단편화(fragmentation)를 나타내는 개략적인 다이어그램을 도시한다.

도 5는 최소 종단간 지연을 위한 파면 병렬 처리 접근법으로의 프레임의 바람직한 단편화를 나타내는 개략적인 다이어그램을 도시한다.

도 6은 비디오 서비스들을 사용하는 대화(conversation)의 시나리오를 나타내는 개략적인 블록 다이어그램을 도시한다.

도 7은 최소 종단간 지연으로의 일반 서브셋들을 갖는 타일들을 위한 인코딩, 전송 및 디코딩의 가능한 시간 스케줄을 개략적으로 도시한다.

도 8은 통상적으로 종단간 지연을 달성하는 타이밍 스케줄을 개략적으로 도시한다.

도 9는 바람직하게는 두 개의 슬라이스로 분할되는, 11x9 코딩 트리블록을 갖는 화상을 도시한다.

도 10은 바람직하게는 세 개의 타일로 분할되는, 13x8 코딩 트리블록을 갖는 화상을 도시한다.

도 11은 시퀀스 파라미터 세트 구문의 일례를 도시한다.

도 12는 화상 파라미터 세트 구문의 일례를 도시한다.

도 13은 슬라이스 헤더 구문을 위한 일례를 도시한다.

도 14는 규칙적인 슬라이스 내로의 파면 병렬 처리 및 저지연 처리를 위한 종속 슬라이스(dependent slice)들 내로의 화상의 분할을 예시한다.

도 15는 화상 파라미터 세트 구문 내의 일 부를 위한 일례들 도시한다.

도 16은 가능한 슬라이스-헤더 구문을 도시한다.

도 17은 정상 슬라이스들(및 종속 슬라이스들)을 위한 코딩 상호의존성을 개략적으로 도시한다.

도 18은 타일들의 저지연 전송을 위한 인코딩을 비교하는 개략적인 다이어그램을 도시한다(종속 슬라이스들을 사용하는 파면 병렬 처리).

도 19는 도 18의 오른쪽 면에 도시된 것과 같이 종속 슬라이스들을 사용하는 파면 병렬 처리를 사용할 때 파이프라인 저지연 전송으로의 바람직한 파면 병렬 처리 코딩을 나타내는 타이밍 스케줄을 도시한다.

도 20은 앵커(anchor)들로서 규칙적인 슬라이스의 사용에 의한 견고성 향상을 나타내는 개략적인 다이어그램을 도시한다.

도 21은 슬라이스 헤더 구문을 위한 또 다른 실시 예를 도시한다.

도 22는 화상 파라미터 세트 구문을 위한 또 다른 실시 예를 도시한다.

도 23은 왼쪽 화상 경계에서 시작하는 경우에 종속 슬라이스를 위한 심볼 확률(symbol probability) 초기화 과정을 나타내는 개략적인 다이어그램을 도시한다.

도 24는 디코더의 개략적인 다이어그램을 도시한다.

도 25는 화상의 코딩 블록들과 슬라이스들로의 분할과 함께 디코더의 블록 다이어그램을 개략적으로 도시한다.

도 26은 인코더의 블록 다이어그램을 개략적으로 도시한다.

도 27은 여기서는 슬라이스 세그먼트들로 불리는, 정상 및 종속 슬라이스들을 개략적으로 도시한다.

도 28은 여기서는 한편으로는 슬라이스 세그먼트들, 다른 한편으로는 타일들로 불리는 정상 및 종속 슬라이스로 분할되는 화상을 개략적으로 도시한다.

도 29는 종속 슬라이스들을 사용하는 콘텍스트 초기화 과정을 나타내는 플로우 다이어그램을 도시한다.

도 30은 종속 슬라이스들을 사용하기 위한 콘텍스트 저장 과정을 나타내는 플로우 다이어그램을 도시한다.

도 31은 파면 병렬 처리 엔트리 지점들을 시그널링하는 서로 다른 가능성을 개략적으로 도시한다.

발명을 실시하기 위한 구체적인 내용

[0020] 다음에서, 각각 병렬 화상 코딩 및 저지연 코딩을 가능하게 하기 위한 오늘날의 개념의 설명으로 구체적인 내용을 시작한다. 두 능력 모두를 가지려고 할 때 발생하는 문제점들이 설명된다. 특히, 다음의 설명에서 나타나는 것과 같이, 지금까지 설명된 파면 병렬 처리 서브스트림 개념은 하나의 슬라이스로 그루핑(grouping)함으로써 파면 병렬 처리 서브스트림들을 전달하기 위한 필요성에 기인하는 저지연을 갖기 위한 바람과 다소 충돌된다. 다음의 실시 예들은 슬라이스 개념을 확장함으로써, 즉 또 다른 형태의 슬라이스를 도입함으로써 훨씬 덜한 지연을 필요하게 하는 애플리케이션들에 적용할 수 있는, 파면 병렬 처리와 같은 병렬 처리 개념들을 제공한다.

[0021] 디스플레이에 대한 포획으로부터 종단간 지연 비디오 지연의 최소화는 비디오 화상회의 등과 같은 적용들에서의 주요 목적 중 하나이다.

[0022] 디지털 비디오 전송을 위한 신호 처리 체인은 카메라, 포획 장치(capturer), 인코더, 캡슐화, 전송, 디멀티플렉서(demultiplexer), 디코더, 렌더러(renderer) 및 디스플레이로 구성된다. 각각의 이러한 단계들은 뒤따르는 단계로의 직렬 전송 전에 이미지 데이터를 버퍼링함으로써 종단간 지연에 기여한다.

[0023] 일부 적용, 예를 들면, 처리되는 대상들을 직접적으로 볼 수 없는, 위험한 영역, 또는 최소 절개 수술에서의 대

상들의 원격 처리는 그러한 지연의 최소화를 필요로 한다.

[0024] 심지어 짧은 지연도 적절한 처리의 심각한 어려움을 야기하거나 또는 비극적인 실수에 이르게 할 수 있다.

[0025] 많은 경우에 있어서, 전체 비디오 프레임은 예를 들면, 인트라 프레임(intra frame) 처리를 허용하기 위한 처리 단계 내에서 버퍼링된다. 일부 단계는 그 다음 단계로 전달되는 패킷들을 형성하기 위하여 데이터를 수집한다. 일반적으로, 지역 처리(local processing)의 요구사항들로부터 야기하는 지연을 위하여 낮은 경계가 존재한다. 이는 아래에 더 상세히 설명되는 각각의 개별 단계를 위하여 분석된다.

[0026] 카메라 내부의 처리는 반드시 인트라-프레임 신호 처리를 필요로 하지 않으며, 따라서 프레임 및 하드웨어 제조사에 의한 일부 디자인 선택에 의해 한정되는, 센서의 적분 시간에 의해 최소 지연이 주어진다. 카메라 출력은 일반적으로 상부 왼쪽 모서리에서 처리를 시작하고 상부 오른쪽 모서리로 이동하며 한 라인씩 하부 오른쪽 모서리로 지속하는 스캔 순서와 관련된다. 결론적으로, 이는 센서로부터 카메라 출력으로 모든 데이터가 전달될 때까지 약 1 프레임 기간이 걸린다.

[0027] 포획 장치는 수신 후에 바로 카메라 데이터를 전달할 수 있으나, 일반적으로 메모리 또는 저장장치로의 데이터 액세스를 최적화하기 위하여 일부 데이터를 버퍼링하고 버스트(burst)들을 발생할 것이다. 게다가, 카메라/포획 장치 및 컴퓨터의 메모리 사이의 연결은 일반적으로 포획된 이미지 데이터의 또 다른 처리(인코딩)를 위한 메모리로의 전달을 위한 비트레이트(bitrate)를 한정한다. 일반적으로, 카메라는 USB 2.0 또는 USB 3.0을 거쳐 연결되나, 항상 인코더로의 이미지 데이터의 부분 전송을 포함할 것이다. 이는 극도의 저지연 시나리오에서 인코더 면 상의 병렬화를 한정하는데, 즉 인코더는 예를 들면, 이미지의 상부로부터 하부로의 래스터-스캔(raster scan)에서, 카메라로부터의 데이터가 이용가능할 때, 가능한 한 빨리 인코딩을 시작하려고 할 것이다.

[0028] 인코더에서, 처리지연의 감소를 위한, 특정 비디오 충실성을 위하여 필요한 데이터 비율과 관련하여, 인코딩 효율을 절충하도록 허용하는 일부 자유도(degree of freedom)가 존재한다.

[0029] 인코더는 그 뒤에 인코딩되는 이미지를 예측하기 위하여 이미 전송된 데이터를 사용한다. 일반적으로, 실제 이미지와 예측 사이의 차이는 예측 없이 필요할 수 있는 것보다 더 적은 비트들로 인코딩될 수 있다. 이러한 예측 값들은 디코더에서 이용할 수 있도록 하기 위하여 필요하며, 따라서 예측은 동일한 이미지의 이전에 디코딩된 부들(인트라-프레임 예측) 또는 이전에 처리된 다른 이미지들(인터(inter)-프레임 예측)을 기초로 한다. 전-HEVC 비디오 인코딩 표준은 위의 이미지 또는 동일한 라인 내, 그러나 왼쪽(이전에 인코딩된)의 일부분, 움직임 벡터 예측 및 엔트로피 코딩(콘텍스트 적응 이진 산술 코딩(CABAC), 이하 CABAC로 표기)만을 사용한다.

[0030] 예측 구조의 최적화에 더하여, 병렬 처리의 영향이 고려될 수 있다. 병렬 처리는 독립적으로 처리될 수 있는 화상 영역의 식별을 필요로 한다. 합리적인 이유를 위하여, 흔히 "타일들"로 불리는, 수평 또는 수직 직사각형과 같은 인접한 영역들이 선택된다. 저지연 제한의 경우에 있어서, 그러한 영역들은 가능한 한 빨리, 포획장치로부터 들어오는 데이터를 메모리로의 병렬화된 코딩을 허용해야만 한다. 래스터-스캔 메모리를 전달할 때, 인코딩을 즉시 시작하기 위하여, 원시 데이터의 수직 파티션들이 이치에 맞는다. 화상을 수직 파티션들로 세분하는(아래의 도면 참조), 그러한 타일들 내부에, 인트라-예측, 움직임 벡터 예측 및 엔트로피 코딩(콘텍스트 기반 적응 산술 코딩)은 합리적인 코딩 효율에 이르게 할 수 있다. 지연을 최소화하기 위하여, 상부로부터 시작하는, 화상의 일부분만이 인코더의 프레임에 전달될 수 있으며, 병렬 처리는 수직 타일들 내에서 시작되어야만 한다.

[0031] 병렬 처리를 허용하는 또 다른 방법은 타일들과 비교될 수 있는, 규칙적인 슬라이스 내의 파면 병렬 처리, 단일 슬라이스 내에 포함된 타일들의 "열"을 사용하는 것이다. 그러한 슬라이스 내의 데이터는 또한 슬라이스 내의, 파면 병렬 처리 서브스트림들을 사용하여 병렬 인코딩될 수 있다. 슬라이스들(900) 및 타일들/파면 병렬 처리 서브스트림들(914)로의 화상 분리가 도 3에 도시된다.

[0032] 따라서, 도 3은 906 또는 914와 같은 병렬 인코딩된 파티션들의 슬라이스 또는 네트워크 전송 세그먼트(단일 네트워크 패킷 또는 다중 네트워크(900) 패킷)로의 지정을 도시한다.

[0033] H.264 또는 HEVC에서 정의된 것과 같이, 인코딩된 데이터의 네트워크 추상 계층(NAL)으로의 캡슐화는 전송 전에 또는 인코딩 과정 동안에 각각의 블록의 식별 및 만일 적용가능하면, 블록들의 재정렬(reordering)을 허용하는 데이터 블록들에 일부 헤더를 추가한다. 표준 사례에서, 어떠한 부가적인 시그널링도 필요하지 않은데, 그 이유는 코딩 요소의 순서가 항상 타일의 위치의 내포된 할당인, 디코딩 순서 내에 존재하거나, 또는 일반적인 코딩 단편이 주어지기 때문이다.

[0034] 만일 저지연 병렬 전송을 위한 부가적인 전송 층으로의 병렬 처리가 고려되면, 즉 전송 층은 저지연 전송을 허

용하기 위하여 타일들을 위한 화상 파티션들을 재정렬할 수 있는데, 이는 그것들이 인코딩됨에 따라 도 4에 도시된 것과 같이 단편들을 보낼 수 있다는 것을 의미한다. 그러한 단편들은 또한 완전히 인코딩된 슬라이스들이 아닐 수 있으며, 슬라이스의 서브셋일 수 있으며, 종속 슬라이스 내에 포함될 수 있다.

[0035] 부가적인 단편들을 생성하는 경우에, 헤더 정보가 일정한 수의 바이트를 추가하기 때문에 큰 데이터 블록들과 함께 가장 높을 수 있는, 효율 및 지연 사이에 균형이 존재하는데, 그 이유는 병렬 인코더들의 큰 데이터 블록들은 전송 이전에 버퍼링될 필요가 있기 때문이다. 전체 지연은 만일 수직 타일들(906)의 인코딩된 표현이 단편이 완전히 인코딩되자마자 전송되는 다수의 단편(916)으로 분리되면 감소될 수 있다. 각각의 단편의 크기는 매크로블록들, 최대 코딩 유닛들과 같은,고정된 이미지 영역 혹은 도 4에 도시된 것과 같이 최대 데이터와 관련하여 결정될 수 있다.

[0036] 따라서, 도 4는 최소 종단간 지연을 위한 타일 코딩 접근법으로의 프레임의 일반적인 단편화를 도시한다.

[0037] 유사하게, 도 5는 최소 종단간 지연을 위한 파면 병렬 처리 코딩 접근법으로의 프레임의 일반적인 단편화를 도시한다.

[0038] 전송은 예를 들면, 만일 전송의 견고성을 증가시키는 순방향 오류 정정(Forward Error Correction)과 같은, 부가적인 블록 지향의 처리가 적용되면 지연을 더 추가할 수 있다. 그밖에, 네트워크 기반구조(라우터(router) 등) 또는 물리적 연결이 지연을 추가할 수 있으며, 이는 일반적으로 연결을 위한 레이턴시(latency)로 알려져 있다. 레이턴시에 더하여, 전송 비트레이트는 비디오 서비스들을 사용하는 도 6에 도시된 것과 같이 대화에서, 파티(Party, a)로부터 파티(b)로 데이터를 전달하기 위한 시간(지연)을 결정한다.

[0039] 만일 인코딩된 데이터 블록들이 제멋대로 전송되면, 지연의 재배열이 고려되어야만 한다. 이에 앞서 디코딩되어야만 하는 다른 데이터 유닛들이 이용가능하면, 디코딩은 데이터 유닛이 도착하자마자 시작할 수 있다.

[0040] 타일들의 경우에 있어서, 타일들 사이에 어떠한 의존성도 없으며, 따라서 타일은 즉시 디코딩될 수 있다. 만일 단편들이 도 4에 도시된 것과 같이 각각의 단편 당 개별 슬라이스들과 같은, 타일로 생산되었으면, 단편들은 각각 그것들의 포함된 최대 코딩 유닛들 또는 코딩 유닛들로 인코딩되자마자 직접적으로 전송될 수 있다.

[0041] 렌더러는 병렬 디코딩 엔진들의 출력들을 어셈블링하고 결합된 화상을 한 라인(line)씩 디스플레이에 전달한다.

[0042] 디스플레이는 어떠한 지연을 반드시 추가하지는 않으나, 실제로 이미지 데이터가 디스플레이되기 전에 실제로 일부 인트라 프레임 처리를 한다. 이는 하드웨어 제조사에 의한 디자인 선택에 달려 있다.

[0043] 요약하면, 최소 종단간 지연을 달성하기 위하여 인코딩, 캡슐화, 전송 및 디코딩 단계들이 영향을 미칠 수 있다. 만일 병렬 처리, 타일들 및 타일들 내의 단편을 사용하면, 총 지연은 도 8에 도시된 것과 같이 이러한 각각의 단계들에서 약 1 프레임 지연을 추가하는 일반적으로 사용되는 처리 체인과 비교하여, 도 7에 도시된 것과 같이 상당히 감소될 수 있다.

[0044] 특히, 도 7은 최소 종단간 지연으로 일반 서브셋을 갖는 타일들을 위한 인코딩, 전송 및 디코딩을 도시하나, 도 8은 일반적으로 달성된 종단간 지연을 도시한다.

[0045] HEVC는 이러한 슬라이스 분할, 타일 분할, 및 다음의 방법으로의 사용을 허용한다:

[0046] 타일: 타일의 트리블록 래스터 스캔 내에 연속적으로 정렬되는, 하나의 행(column)과 하나의 열에서 공동 발생하는 정수(integer number)의 트리블록. 화상 내의 타일들은 화상의 타일 래스터 스캔 내에 연속적으로 정렬된다. 슬라이스가 타일의 트리블록 래스터 스캔 내에서 연속적이나, 이러한 타일블록들은 화상의 트리블록 래스터 스캔 내에서 반드시 연속적이지는 않다.

[0047] 슬라이스: 래스터 스캔 내에 연속적으로 정렬되는 정수의 트리블록들. 슬라이스들로의 각각의 화상의 세분화가 분할이다. 트리블록 주소들은 슬라이스 내의 제 1 트리블록 어드레스로부터 도출된다.

[0048] 래스터 스캔: 1차원 패턴 내의 제 1 엔트리들이 왼쪽으로부터 오른쪽으로 스캐닝되는 2차원 패턴의 제 1 상부 열로부터 존재하고, 유사하게 각각 왼쪽으로부터 오른쪽으로 스캐닝되는 패턴의 제 2, 제 3 등의 열이 뒤따르는 것과 같은 직사각형 2차원 패턴의 1차원 패턴으로의 매핑.

[0049] 트리블록: 4개의 샘플 어레이를 갖는 화상의 루마 샘플(luma sample)들의 NxN 블록 및 크로마 샘플(chroma sample)의 두 개의 상응하는 블록, 혹은 3개의 개별 색 평면을 사용하여 코딩되는 모노크롬 화상 또는 화상의 샘플들의 NxN 블록. 트리블록들로의 슬라이스의 세분이 분할이다.

[0050] 분할: 세트의 각각의 요소가 정확하게 서브셋들의 하나인 것과 같이 하나의 세트의 서브셋들로의 세분.

[0051] 쿼드트리: 부모 노드(parent node)가 4개의 자식 노드로 분할될 수 있는 트리. 자식 노드는 4개의 자식 노드로의 또 다른 분할을 위한 부모 노드가 될 수 있다.

[0052] 다음으로, 화상들, 슬라이스들 및 타일들의 공간 세분화가 설명된다. 특히, 다음의 설명은 화상이 어떻게 슬라이스들, 타일들 및 코딩 트리블록들로 분할되는지를 명시한다. 화상들은 슬라이스들과 타일들로 세분된다. 슬라이스는 코딩 트리블록들의 시퀀스이다. 유사하게, 타일은 코딩 트리블록들의 시퀀스이다.

[0053] 샘플들은 코딩 트리블록의 유닛들 내에서 처리된다. 샘플들 내의 각각의 트리블록을 위한 루마 어레이 크기는 CtbSize이다, 각각의 코딩 트리블록을 위한 크로마 어레이의 폭 및 높이는 각각, CtbWidthC 및 CtbHeightC이다. 예를 들면, 화상은 그 다음의 도면에 도시된 것과 같이 두 개의 슬라이스로 세분될 수 있다. 또 다른 실시 예에서와 같이, 화상은 그 다음 두 번째 도면에 도시된 것과 같이 세 개의 타일로 세분될 수 있다.

[0054] 슬라이스들과 다르게, 타일들은 항상 직사각형이고 항상 코딩 트리블록 래스터 스캔 내에 정수의 코딩 트리블록을 포함한다. 타일은 하나 이상의 슬라이스 내에 포함된 코딩 트리블록들로 구성될 수 있다. 유사하게 슬라이스는 하나 이상의 슬라이스 내에 포함된 코딩 트리블록들로 구성될 수 있다.

[0055] 도 9는 두 개의 슬라이스(900a, b)로 분할되는 13x8 코딩 트리블록들(918)을 갖는 화상(898)을 도시한다.

[0056] 도 10은 세 개의 타일로 분할되는 13x8 코딩 트리블록들(918)을 갖는 화상을 도시한다.

[0057] 각각의 코딩(898) 트리블록(918)은 인트라 또는 인터 예측 및 변환 코딩을 위한 블록 크기들을 식별하기 위한 파티션 시그널링이 지정된다. 분할은 재귀적(recursive) 쿼드트리 분할이다. 쿼드트리의 루트는 코딩 트리블록과 관련된다. 쿼드트리는 리프(leaf)가 도달할 때까지 분열되는데, 이는 코딩 블록으로서 언급된다. 코딩 블록은 두 개의 트리, 예측 트리와 변환 트리의 루트 노드(root node)이다.

[0058] 예측 트리는 예측 블록들의 위치와 크기를 지정한다. 예측 블록들 및 관련 예측 데이터는 예측 유닛으로서 언급된다.

[0059] 도 11은 바람직한 시퀀스 파라미터 세트 원시 바이트 시퀀스 페이로드(RBSP, 이하 RBSP로 표기) 구문을 도시한다.

[0060] 변환 트리는 변환 블록들의 위치와 크기를 지정한다. 변환 블록들 및 관련 변환 데이터는 변환 유닛으로서 언급된다.

[0061] 루마 및 크로마를 위한 분할 정보는 예측 트리를 위하여 동일하며 변환 트리를 위하여 동일하거나 또는 동일하지 않을 수 있다.

[0062] 코딩 블록, 관련 코딩 데이터 및 관련 예측과 변환 유닛은 함께 코딩 유닛을 형성한다.

[0063] 타일 스캔 순서에 대한 코딩 트리블록 래스터 순서에서의 코딩 트리블록 주소의 대화를 위한 과정은 다음과 같을 수 있다:

[0064] 이러한 과정의 출력들은 다음과 같다:

[0065] - 0 내지 PicHeightInCtbs*PicWidthInCtbs - 1의 범위의 ctbAddrRS를 갖는, 어레이 CtbAddrTS[ctbAddrRS]

[0066] - 0 내지 PicHeightInCtbs*PicWidthInCtbs - 1의 범위의 ctbAddrTS를 갖는, 어레이 TileId[ctbAddrTS]

[0067] CtbAddrTS[]의 어레이는 다음과 같이 도출된다:

```
for( ctbAddrRS = 0; ctbAddrRS < PicHeightInCtbs * PicWidthInCtbs; ctbAddrRS++ ) {
    tbX = ctbAddrRS % PicWidthInCtbs
    tbY = ctbAddrRS / PicWidthInCtbs
    for( j = 0; j <= num_tile_columns_minus1; j++ )
        if( tbX < ColBd[ j + 1 ] )
            tileX = j
    for( i = 0; i <= num_tile_rows_minus1; i++ )
        if( tbY < RowBd[ i + 1 ] )
            tileY = i
    CtbAddrTS[ ctbAddrRS ] = ctbAddrRS - tbX
            for( i = 0; i < tileX; i++ )
        ctbAddrTS += RowHeight[ tileY ] * ColumnWidth[ i ]
    CtbAddrTS[ ctbAddrRS ] += ( tbY - RowBd[ tileY ] ) * ColumnWidth[ tileY ] + tbX - ColBd[ tileX ]
}
```

[0068]

[0069] 어레이 CtbAddrTS[]는 다음과 같이 도출된다:

```
for( j = 0, tileId = 0; j <= num_tile_columns_minus1; j++ )
    for( i = 0; i <= num_tile_rows_minus1; i++, tileId++ )
        for( y = RowBd[ j ]; y < RowBd[ j + 1 ]; y++ )
            for( x = ColBd[ i ]; x < ColBd[ i + 1 ]; x++ )
                TileId[ CtbAddrTS[ y*PicWidthInCtbs + x ] ] = tileId
```

[0070]

[0071] 어레이 TileId[]는 다음과 같이 도출된다:

```
for( i = 0; i <= num_tile_columns_minus1; i++ ) {
    if( uniform_spacing_flag )
        ColumnWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbs ) / ( num_tile_columns_minus1 + 1 ) -
                           ( i * PicWidthInCtbs ) / ( num_tile_columns_minus1 + 1 )
    else
        ColumnWidth[ i ] = column_width[ i ]
    ColumnWidthInLumaSamples[ i ] = ColumnWidth[ i ] << Log2CtbSize
}
```

[0072]

[0073] 상응하는, 바람직한 구문이 도 11, 12 및 13에 도시되는데, 도 12는 바람직한 화상 파라미터 세트 RBSP 구문을 갖는다. 도 13은 바람직한 슬라이스 헤더 구문을 도시한다.

[0074] 구문 실시 예에서, 다음의 시맨틱(semantic)이 적용될 수 있다:

[0075] 1과 동일한 entropy_slice_flag는 존재하지 않는 슬라이스 헤더 구문 요소들의 값이 선행 슬라이스 내의 슬라이스 헤더 구문 요소들의 값과 동일하도록 추론된다는 것을 지정하는데, 선행 슬라이스는 위치(SliceCtbAddrRS-1)를 갖는 코딩 트리블록을 포함하는 슬라이스로서 정의된다. SliceCtbAddrRS가 0과 동일할 때 entropy_slice_flag는 0과 동일해야만 한다.

[0076] 0과 동일한 tiles_or_entropy_coding_sync_idc는 코딩된 비디오 시퀀스 내의 각각의 화상 내에 하나의 타일만이 존재하고, 코딩 트리블록들의 열의 제 1 코딩 트리블록을 디코딩하기 전에 콘텍스트 변수들을 위한 어떠한 특정 화 과정도 적용되지 않는다는 것을 지정한다.

[0077] 1과 동일한 tiles_or_entropy_coding_sync_idc는 코딩된 비디오 시퀀스 내의 각각의 화상 내에 하나 이상의 타일이 존재하고, 코딩 트리블록들의 열의 제 1 코딩 트리블록을 디코딩하기 전에 콘텍스트 변수들을 위한 어떠한 특정 동기화 과정도 적용되지 않는다는 것을 지정한다.

[0078] 2와 동일한 tiles_or_entropy_coding_sync_idc는 코딩된 비디오 시퀀스 내의 각각의 화상 내에 하나 이상의 타일이 존재하고, 코딩 트리블록들의 열의 제 1 코딩 트리블록을 디코딩하기 전에 콘텍스트 변수들을 위한 특정 동기화 과정이 적용되며, 코딩 트리블록들의 열의 두 개의 코딩 트리블록을 디코딩한 후에 콘텍스트 변수들을 위한 특정 기억 과정이 적용된다는 것을 지정한다.

[0079] tiles_or_entropy_coding_sync_idc의 값은 0 내지 2의 범위 내에 존재하여야만 한다.

[0080] num_tile_columns_minus1 plus 1은 화상을 분할하는 타일 행들의 수를 지정한다.

[0081] num_tile_rows_minus1 plus 1은 화상을 분할하는 타일 열들의 수를 지정한다. num_tile_columns_minus1이 0과 동일할 때, num_tile_rows_minus1은 0과 동일해서는 안 된다.

[0082] 다음의 조건 중 하나 또는 둘 모두는 각각의 슬라이스 및 타일을 위하여 실현되어야만 한다:

[0083] - 슬라이스 내의 모든 인코딩된 블록은 동일한 타일에 속한다.

[0084] - 타일 내의 모든 인코딩된 블록은 동일한 슬라이스에 속한다.

[0085] 주의 - 동일한 화상 내에서, 다수의 타일을 포함하는 슬라이스들 및 다수의 슬라이스를 포함하는 타일들 모두가 존재할 수 있다.

[0086] 1과 동일한 uniform_spacing_flag는 행 경계들과 유사한 열 경계들이 화상을 가로질러 균일하게 분포되는 것을 지정한다. 0과 동일한 uniform_spacing_flag는 행 경계들과 유사한 열 경계들이 화상을 가로질러 균일하게 분포되지 않으나, 구문 요소들 column_width[i] 및 row_height[i]를 사용하여 분명하게 시그널링되는 것을 지정한다.

[0087] column_width[i]는 코딩 트리블록들의 유닛들 내의 i번째 타일 행의 폭을 지정한다.

[0088] row_height[i]는 코딩 트리블록들의 유닛들 내의 i번째 타일 열의 높이를 지정한다. 코딩 트리블록들의 유닛들 내의 i번째 타일 행의 폭을 지정하는, Column Width[i]의 값들 및 루마 샘플들 내의 i번째 타일 행의 높이를 지정하는, Column WidthInLumaSamples[i]의 값들은 다음과 같이 도출된다:

```
for( i = 0; i <= num_tile_rows_minus1; i++ )
    if( uniform_spacing_flag )
      RowHeight[ i ] = ( ( i + 1 ) * PicHeightInCtbs ) / ( num_tile_rows_minus1 + 1 ) –
                       ( i * PicHeightInCtbs ) / ( num_tile_rows_minus1 + 1)
    else
      RowHeight[ i ] = row_height[ i ]
```

[0089]

[0090] 코딩 트리블록들의 유닛들 내의 i번째 타일 열의 높이를 지정하는, RowHeight[i]의 값들은 다음과 같이 도출된다:

```
for( i = 0; i <= num_tile_rows_minus1; i++ )
    if( uniform_spacing_flag )
      RowHeight[ i ] = ( ( i + 1 ) * PicHeightInCtbs ) / ( num_tile_rows_minus1 + 1 ) –
                       ( i * PicHeightInCtbs ) / ( num_tile_rows_minus1 + 1)
    else
      RowHeight[ i ] = row_height[ i ]
```

[0091]

[0092] 코딩 트리블록들의 유닛들 내의 i번째 타일 행의 왼쪽 행 경계의 위치를 지정하는, ColBd[1]의 값들은 다음과 같이 도출된다:

```
for( RowBd[ 0 ] = 0, i = 0; i <= num_tile_rows_minus1; i++ )
   RowBd[ i + 1 ] = RowBd[ i ] + RowHeight[ i ]
```

[0093]

[0094] 코딩 트리블록들의 유닛들 내의 i번째 타일 열의 상부 열 경계의 위치를 지정하는, RowBd[1]의 값들은 다음과 같이 도출된다:

```
for( RowBd[ 0 ] = 0, i = 0; i <= num_tile_rows_minus1; i++ )
   RowBd[ i + 1 ] = RowBd[ i ] + RowHeight[ i ]
```

[0095]

[0096] num_substreams_minus1 plus 1은 tiles_or_entropy_coding_sync_idc가 2와 동일할 때 슬라이스 내에 포함된 서브셋들의 최대 수를 지정한다. 존재하지 않을 때, num_substreams_minus1의 값은 0과 동일한 것으로 추론된다.

[0097] num_entry_point_offsets는 슬라이스 헤더 내의 entry_point_offset[i] 구문 요소들의 수를 지정한다. tiles_or_entropy_coding_sync_idc가 1과 동일할 때, num_entry_point_offsets의 값은 0 내지 (num-tile_columns_minus1 + 1) * (num_tile_rows_minus1 + 1) - 1의 범위 내에 존재하여야만 한다. tiles_or_entropy_coding_sync_idc가 2와 동일할 때, num_entry_point_offsets의 값은 0 내지 num-substreams_minus1의 범위 내에 존재하여야만 한다. 존재하지 않을 때, num_entry_point_offsets의 값은 0과 동일한 것으로 추론된다.

[0098] offset_len_minu1 plus 1은 entry_point_offset[i] 구문 요소들의 비트들 내의 길이를 지정한다.

[0099] entry_point_offset[i]는 바이트들 내의 i번째 엔트리 지점 오프셋을 지정하며, offset_len_minu1 plus 1 비트들에 의해 표현되어야만 한다. 코딩된 슬라이스 네트워크 추상 계층 유닛은 0부터 num_entry_point_offsets까지의 범위의 지수 값을 갖는, num_entry_point_offsets + 1 서브셋들로 구성된다. 서브셋 0은 코딩된 슬라이스 네트워크 추상 계층 유닛의 바이트 0 내지 entry_point_offset[0]-1로 구성되고, 1 내지 num_entry_point_offsets-1의 범위 내의 k를 갖는, 서브셋 k는 바이트들 entry_point_offset[k-1] 내지 entry_point_offset[k]+entry_point_offset[k-1]-1로 구성되며, 마지막 서브셋(num_entry_point_offsets와 동일한 서브셋 지수를 갖는)은 코딩된 슬라이스 네트워크 추상 계층 유닛의 나머지 바이트들로 구성된다.

[0100] 주의- 네트워크 추상 계층 유닛 헤더 및 코딩된 슬라이스 네트워크 추상 계층 유닛 내의 슬라이스 헤더는 항상 서브셋 0 내에 포함된다.

[0101] tiles_or_entropy_coding_sync_idc가 1과 동일하고 num_entry_point_offsets가 0보다 클 때, 각각의 서브셋은 하나 또는 다수의 완전한 타일의 모든 코딩된 비트를 포함하여야만 하고 서브셋들의 수는 슬라이스 내의 타일들의 수와 동일하거나 또는 작아야만 한다.

[0102] tiles_or_entropy_coding_sync_idc가 2와 동일하고 num_entry_point_offsets가 0보다 클 때, 각각의 모든 가능한 k 값들을 위한 서브셋 k는 현재 비트스트림 포인터 k를 위한 초기화 과정 동안에 사용되도록 모든 비트를 포함하여야만 한다.

[0103] 슬라이스 데이터 시맨틱과 관련하여, 다음이 적용될 수 있다:

[0104] 0과 동일한 end_of_slice_flag는 또 다른 매크로블록이 슬라이스 내에 따르는 것을 지정한다. 1과 동일한 end_of_slice_flag는 슬라이스의 끝을 지정하고 어떠한 다른 매크로블록도 따르지 않는 것을 지정한다.

[0105] entry_point_marker_two_3bytes는 0x000002와 동일한 3 바이트의 고정된 값 시퀀스이다. 이러한 구문 요소는 엔트리 마커 프리픽스(entry marker prefix)로 불린다.

[0106] tile_idx_minus_1은 래스터 스캔 순서 내의 TileID를 지정한다. 화상 내의 첫 번째 타일은 0의 TileID를 가져야만 한다. tile_idx_minus_1의 값은 0 내지 (num_tile_columns_minus1+1)*(num_tile_rows_minu1+1)-1의 범위 내에 존재하여야만 한다.

[0107] 슬라이스 데이터를 위한 CABAC 파싱(parsing) 과정은 다음과 같다:

[0108] 이러한 과정은 기술자(descriptor) ae(v)를 갖는 구문 요소를 파싱할 때 적용된다.

[0109] 이러한 과정에 대한 입력들은 구문 요소의 값 및 이전에 파싱된 구문 요소들의 값들의 요구(request)이다.

[0110] 이러한 과정의 출력은 구문 요소의 값이다.

[0111] 슬라이스의 슬라이스 데이터의 파싱을 시작할 때, CABAC 파싱 과정의 초기화 과정이 적용된다. tiles_orentropy_coding_sync_idc가 2와 동일하고 num_substreams_minus1+이 0보다 클 때, 나중을 위하여 현재 비트스트림 포인터(bitstream pointer) 편차를 사용하기 위하여 비트스트림 포인트 테이블을 지정하는 num_substreams_minus1 + 1을 갖는, 매핑 테이블 BitStreamTable은 다음과 같이 도출된다.

[0112] - 비트스트림 포인터를 포함하기 위하여 BitStreamTable[0]이 초기화된다.

[0113] - 0보다 크고 num-substreams_minus1 + 1보다 작은 모든 지수(i)를 위하여, BitStreamTable[i]는 BitStreamTable[i-1] 이후의 entry_point_offset[i]에 대한 비트스트림 포인터를 포함한다.

[0114] 현재의 비트스트림 포인터는 BitStreamTable[0]으로 설정된다.

[0115] 공간적 이웃 블록(T)을 포함하는 코딩 트리블록의 최소 코딩 블록 주소(ctbMinCbAddrT)는 예를 들면, 다음과 같은 현재 코딩 트리블록의 상부 왼쪽 루마 샘플의 위치(x0, y0)를 사용하여 도출된다.

[0116] x= x0 + 2 ≪ Log2CtbSize - 1

[0117] y = y0 - 1

[0118] ctbMinCbAddrT = MinCbAddrZS[x ≫ Log2MinCbSize][y ≫ Log2MinCbSize]

[0119] 변수(availableFlagT)는 입력으로서 ctbMinCbAddrT를 갖는 적절한 코딩 블록 이용가능성 도출 과정을 적용함으

로써 획득된다.

[0120] 코딩 트리의 파싱을 시작하고 tiles_or_entropy_coding_sync_idc이 2와 동일하고 num_substreams_minus1이 0보다 클 때, 다음이 적용된다.

[0121] - 만일 CtbAddrRS%PicWidthInCtbs가 0과 동일하면, 다음이 적용된다.

[0122] - availableFlagT가 1과 동일할 때, 하위절(subclause) "콘텍스트 변수들을 위한 동기화 과정"에 지정된 것과 같이 CABAC 파싱 과정의 동기화 과정이 적용된다.

[0123] - 종료 이전에 이진 결정을 위한 디코딩 과정이 적용되고 산술 디코딩 엔진을 위한 초기화 과정이 뒤따른다.

[0124] - 현재 비트스트림 포인터는 다음과 같이 도출되는 지수(i)를 갖는 BitStreamTable[i]를 나타내도록 설정된다.

[0125] i = (CtbAddrRS/PicWidthInCtbs)%(num_substreams_minus1 + 1)

[0126] - 그렇지 않으면, 만일 CtbAddrRS%PicWidthInCtbs가 2와 동일하면, 하위절 "콘텍스트 변수들을 위한 기억 과정"에 지정된 것과 같이 CABAC 파싱 과정의 기억 과정이 적용된다.

[0127] 초기화 과정은 다음과 같을 수 있다:

[0128] 이러한 과정의 출력들은 초기화된 CABAC 내부 변수들이다.

[0129] 슬라이스의 슬라이스 데이터의 파싱을 시작할 때 또는 코딩 트리의 데이터의 파싱을 시작하고 코딩 트리가 타일 내의 첫 번째 코딩 트리일 때 그것들의 특별한 과정들이 적용된다.

[0130] 콘텍스트 변수들을 위한 기억 과정은 다음과 같을 수 있다:

[0131] 이러한 과정의 입력들은 ctxIdx에 의해 인덱싱되는 CABAC 콘텍스트 변수들이다.

[0132] 이러한 과정의 출력은 엔드-오브-슬라이스(end-of-slice) 플래그를 제외하고 구문 요소들에 지정되는 콘텍스트 변수들의 초기화 과정에서 사용되는 변수들(m 및 n)의 값들을 포함하는 변수들(TableStateSync 및 TableMPSSync)이다.

[0133] 각각의 콘텍스트 변수를 위하여, 테이블들(TableStateSync 및 TableMPSSync)의 상응하는 엔트리들(n 및 m)이 상응하는 pStateIdx 및 valMPS로 초기화된다.

[0134] 콘텍스트 변수들을 위한 동기화 과정은 다음과 같을 수 있다:

[0135] 이러한 과정의 입력들은 엔드-오브-슬라이스 플래그를 제외하고 구문 요소들에 지정되는 콘텍스트 변수들의 기억 과정에서 사용되는 변수들(n 및 m)의 값들을 포함하는 변수들(TableStateSync 및 TableMPSSync)이다.

[0136] 이러한 과정의 출력들은 ctxIdx에 의해 인덱싱되는 CABAC 콘텍스트 변수들이다.

[0137] 각각의 콘텍스트 변수를 위하여, 상응하는 콘텍스트 변수들(pStateIdx 및 valMPS)은 테이블들(TableStateSync 및 TableMPSSync)의 상응하는 엔트리들(n 및 m)로 초기화된다.

[0138] 다음에서, 파면 병렬 처리를 사용하는 저지연 코딩 및 전송이 설명된다. 특히, 아래의 설명은 도 7에 도시된 것과 같은 저지연 전송이 어떻게 또한 파면 병렬 처리에 적용될 수 있는지를 나타낸다.

[0139] 우선, 전체 화상의 완성 이전에, 화상의 서브셋이 보내질 수 있는 것이 중요하다. 정상적으로는, 이미 도 5에 도시된 것과 같이, 이는 슬라이스들을 사용하여 달성될 수 있다.

[0140] 다음의 도면들에서 도시되는 것과 같이 타일들과 비교하여 지연을 감소시키기 위하여, 최대 코딩 유닛들의 열당 단일 파면 병렬 처리 서브스트림을 적용하고 또한 각각의 그러한 열들의 개별 전송을 허용하기 위한 필요성이 존재한다. 코딩 효율을 높게 유지하기 위하여, 각각의 열/서브스트림 당 슬라이스들은 사용될 수 없다. 따라서, 아래의, 그 다음이 섹션에서 정의되는 것과 같이 이른바 종속 슬라이스가 도입된다. 이러한 슬라이스는 예를 들면, 완전한 HEVC 슬라이스 헤더의 모든 필드를 갖지 않으나, 엔트로피 슬라이스들을 위하여 사용되는 필드들을 갖는다. 게다가, 열들 사이의 CABAC의 정지(break)를 끄기 위한 스위치가 존재할 수 있다. 파면 병렬 처리의 경우에 있어서, CABAC 콘텍스트(도 14의 화살표들)의 사용 및 열들의 예측은 타일에 대한 파면 병렬 처리의 코딩 효율 이득을 유지하도록 허용되어야만 한다.

[0141] 특히, 도 14는 규칙적인 슬라이스(900, reg.SL)로의 파면 병렬 처리 및 종속 슬라이스들(920, OS)로의 저지연

처리를 위한 화상(10)을 예시한다.

[0142] 현재의 새로운 HEVC 표준은 슬라이스들과 관련하여 두 가지 형태의 분할을 제공한다. 규칙적인(정상) 슬라이스 및 엔트로피 슬라이스가 존재한다. 규칙적인 슬라이스는 슬라이스 경계들 상의 블록화 제거 필터 과정에 기인하여 이용가능할 수 있는 일부 의존성을 제외하고 완전히 독립된 화상 파티션이다. 엔트로피 슬라이스는 또한 엔트로피 코딩과 관련해서만 독립적이다. 도 14의 개념은 슬라이싱 개념을 일반화하는 것이다. 따라서 새로운 HEVC 표준은 두 가지 일반적인 슬라이스의 형태: 독립(규칙적인) 또는 종속을 제공해야만 한다. 따라서, 새로운 형태의 슬라이스, 종속 슬라이스가 도입된다.

[0143] 종속 슬라이스는 이전의 슬라이스에 대한 의존성을 갖는 슬라이스이다. 의존성은 엔트로피 디코딩 과정 및/또는 픽셀 재구성 과정에서 슬라이스들 사이에서 사용될 수 있는 특정 데이터이다.

[0144] 도 14에서 종속 슬라이스들의 개념이 바람직하게 표현된다. 화상은 예를 들면, 항상 규칙적인 슬라이스로 시작한다. 이러한 개념에서 규칙적인 슬라이스 행동은 약간 변경된다는 것에 유의하여야 한다. 일반적으로, H.264/AVC 또는 HEVC 같은 표준들에서, 규칙적인 슬라이스는 완전히 독립적인 파티션이고 블록화 제거 필터 과정을 위한 일부 데이터를 제외하고는 디코딩 후에 어떠한 데이터도 유지해서는 안 된다. 그러나 다음의 종속 슬라이스(920)는 위의 슬라이스, 여기서는 첫 번째 열 내의 규칙적인 슬라이스(900)의 데이터를 참조함으로써만 가능하다. 이를 달성하기 위하여, 규칙적인 슬라이스(900)는 마지막 코딩 유닛-열의 데이터를 유지해야만 한다.

[0145] 이러한 데이터는 다음을 포함한다:

[0146] - CABAC 코딩 엔진 데이터(종속 슬라이스의 엔트로피 디코딩 과정이 초기화될 수 있는 하나의 코딩 유닛의 콘텍스트 모델 상태들),

[0147] - 종속 코딩 유닛들의 규칙적인 CABAC 디코딩 과정을 위한 코딩 유닛들의 모든 코딩된 구문 요소,

[0148] - 인트라 및 움직임 벡터 예측의 데이터.

[0149] 그 결과 각각의 종속 슬라이스(920)는 동일한 절차(동일한 화상 내의 다가올 종속 슬라이스를 위한 데이터의 유지)를 수행하여야만 한다.

[0150] 실제로, 이러한 부가적인 단계들이 문제가 되어서는 안 되는데, 그 이유는 디코딩 과정이 일반적으로 항상 구문 요소들 같은 일부 데이터를 저장하도록 강요받기 때문이다.

[0151] 아래의 섹션들에서 종속 슬라이스들의 개념을 가능하게 하는데 필요한 HEVC 표준 구문을 위한 가능한 변화들이 표현된다.

[0152] 예를 들면, 도 5는 화상 파라미터 세트 RBSP 구문의 가능한 변화를 나타낸다.

[0153] 종속 슬라이스들을 위한 화상 파라미터 세트 시멘틱은 다음과 같을 수 있다:

[0154] 1과 동일한 **dependent_slices_present_flag**는 화상이 종속 슬라이스들을 포함하며 각각의 (규칙적인 또는 종속) 슬라이스의 디코딩 과정이 또한 규칙적인 슬라이스를 뒤따를 수 있는 종속 슬라이스일 수 있는 그 다음의 슬라이스를 위하여 엔트로피 디코딩의 상태들, 및 인트라 및 움직임 벡터 예측을 저장하는 것을 지정한다. 그 다음의 종속 슬라이스는 그러한 저장된 데이터를 참조할 수있다.

[0155] 도 16은 HEVC의 현재 상태에 대한 변화들을 갖는 가능한 slice_header 구문을 도시한다.

[0156] 1과 동일한 **dependent_slice_flag**는 존재하지 않는 슬라이스 헤더 구문 요소들의 값이 선행(규칙적인) 슬라이스 내의 슬라이스 헤더 구문 요소들의 값과 동일하도록 추론되는 것을 지정하며, 선행 슬라이스는 위치(SliceCtbAddrRS-1)를 갖는 코딩 트리블록을 포함하는 슬라이스로서 정의되고, SliceCtbAddrRS가 0과 동일할 때 dependent_slice_flag는 0과 동일해야만 한다.

[0157] 1과 동일한 **no_cabac_reset_flag**는 이전에 디코딩된 슬라이스(및 초기 값들을 갖지 않는)의 저장된 상태로부터의 CABAC 초기화를 지정한다. 그렇지 않으면, 즉, 만일 0이면, 이전에 디코딩된 슬라이스의 어떠한 상태와 관계없는, 즉 초기 값들을 갖는 CABAC 초기화가 지정된다.

[0158] 1과 동일한 **last_ctb_cabac_init_flag**는 이전에 디코딩된 슬라이스(예를 들면, 항상 1과 동일한 타일들을 위하여)마지막 코딩된 트리블록이 저장된 상태로부터의 CABAC 초기화를 지정한다. 그렇지 않으면(0과 동일하면), 만일 현재 슬라이스의 첫 번째 코딩된 트리블록이 열 내의 첫 번째 코딩된 트리블록이면(즉, 파면 병렬 처리 방식), 초기화 데이터는 이전에 디코딩된 슬라이스의 마지막(이웃하는) ctb-row의 두 번째 코딩된 트리블록의

저장된 상태로부터 참조되며, 그렇지 않으면 CABAC 초기화는 이전에 코딩된 슬라이스의 마지막 코딩된 트리블록의 저장된 상태로부터 실행된다.

[0159] 종속 슬라이스들 및 다른 분할 전략들(유용한)의 비교가 아래에 제공된다.

[0160] 도 17에서, 정상 및 종속 슬라이스들 사이의 차이점이 도시된다.

[0161] 도 18과 관련하여 도시된 것과 같이 종속 슬라이스들(DS) 내의 파면 병렬처리 서브스트림들의 가능한 코딩 및 전송은 타일들(왼쪽) 및 파면 병렬 처리/종속 슬라이스의 저지연 전송을 위한 인코딩을 비교한다. 도 18에서 굵게 연속적으로 도시된 선은 파면 병렬 처리 열의 인코딩이 단일 타일의 인코딩과 동일한 시간이 걸릴 때 두 가지 방법을 위한 시간의 동일한 시간 지점을 나타낸다. 코딩 의존성에 기인하여, 파면 병렬 처리의 첫 번째 라인만이 준비되며, 결국 타일들은 인코딩되었다. 그러나 종속 슬라이스를 사용하여 접근법은 일단 인코딩되면 첫 번째 열을 보내기 위하여 파면 병렬 처리 접근법을 허용한다.

[0162] 이는 초기의 파면 병렬 처리 서브스트림 지정들과는 다르며, "서브스트림"은 동일한 디코더 스레드(decoder thread), 즉, 동일한 코어/프로세서에 의해 디코딩되는 파면 병렬 처리가 되도록 코딩 유닛 열들의 연결로서의 파면 병렬 처리를 위하여 정의된다. 비록 이전에 열 당 서브스트림 및 엔트로피 슬라이스 당 서브스트림이 또한 가능하였을 수 있으나, 엔트로피 슬라이스는 엔트로피 코딩 의존성을 깨뜨리고 따라서 낮은 코딩 효율을 갖는데, 즉 파면 병렬 처리 효율 이득이 손실된다.

[0163] 부가적으로, 두 접근법 사이의 지연 차이는 도 19에 도시된 것과 같은 전송이라고 가정하면, 실제로 낮을 수 있다. 특히, 도 19는 파이프라인 방식의 저지연 전송을 갖는 파면 병렬 처리 코딩을 도시한다.

[0164] 도 18의 파면 병렬 처리 접근법에서 종속 슬라이스 #1.1의 후자의 두 개의 코딩 유닛의 인코딩이 첫 번째 열 SL #1의 전송보다 길게 걸리지 않는다고 가정하면, 저지연의 경우에서 타일들과 파면 병렬 처리 사이에 어떠한 차이도 존재하지 않는다. 그러나 파면 병렬 처리/종속 슬라이스의 코딩 효율은 타일 개념을 능가한다.

[0165] 파면 병렬 처리 저지연 방식을 위한 견고성을 증가시키기 위하여, 도 20은 앵커들과 같은 규칙적인 슬라이스들(RS)을 사용함으로써 견고성 향상이 달성되는 것을 도시한다. 도 20에 도시된 화상에서 (규칙적인) 슬라이스(RS)는 종속 슬라이스(DS)를 뒤따른다. 여기서, (규칙적인) 슬라이스는 선행 슬라이스들에 대한 의존성을 깨뜨리기 위한 앵커로서 작용하며, 따라서 (규칙적인) 슬라이스의 그러한 삽입 지점에서 더한 견고성이 제공된다. 원칙적으로, 이는 어쨌든 (규칙적인) 슬라이스를 삽입하는 것과 다르지 않다.

[0166] 종속 슬라이스들의 개념은 또한 다음과 같이 구현될 수 있다:

[0167] 여기서, 도 21은 가능한 슬라이스 헤더 구문을 도시한다.

[0168] 슬라이스 헤더 시멘틱은 다음과 같다:

[0169] 1과 동일한 **dependent_slice_flag**는 존재하지 않는 각각의 슬라이스 헤더 구문 요소의 값이 코딩 트리블록 주소가 SliceCtbAddrRS-1인 코딩 트리블록을 포함하는 선행 슬라이스 내의 상응하는 슬라이스 헤더 구문 요소의 값과 동일하도록 추론되는 것을 지정한다.

[0170] **slice_address**는 슬라이스가 시작하는 입상 해상도 내의 슬라이스 주소를 지정한다. slice_address 구문 요소의 길이는 (Ceil(Log2(PicWidthInCtbs *PicHeightInCtbs))+SliceGranularity) 비트이다.

[0171] 슬라이스가 코딩 트리블록 래스터 스캔 순서로 시작하는 코딩 트리블록을 지정하는, 변수 SliceCtbAddrRS는 다음과 같이 도출된다.

[0172] SliceCtbAddrRS = (slice_address ≫ SliceGranularity)

[0173] Z-스캔 순서로 최소 코딩 블록 입상도로 슬라이스 내의 첫 번째 코딩 블록의 주소를 지정하는, 변수 SliceCtbAddrZS는 다음과 같이 도출된다.

[0174] SliceCbAddrZS = slice_address

[0175] ≪ ((log2_diff_max_min_coding_block_size-SliceGranularity) ≪ 1).

[0176] 슬라이스 디코딩은 가능한 가장 큰 코딩 유닛과 함께, 또는 다른 관점에서, 슬라이스 시작 좌표에서 CTU로 시작한다.

[0177] **first_slice_in_pic_flag**는 슬라이스가 화상의 첫 번째 슬라이스인지를 나타낸다. 만일

first_slice_in_pic_flag가 1과 동일하면, 변수들 SliceCbAddrZS 및 SliceCtbAddrRS는 모두 0으로 설정되고 디코딩은 화상 내의 첫 번째 코딩 트리블록으로 시작한다.

[0178] **pic_parameter_set_id**는 사용중인 화상 파라미터 세트를 지정한다. pic_parameter_set_id의 값은 0 내지 255의 범위 내에 존재하여야만 한다.

[0179] **num_entry_point_offsets**는 슬라이스 헤더 내의 entry_point_offset[i] 구문 요소들의 수를 지정한다. tiles_or_entropy_coding_sync_idc가 1과 동일할 때, num_entry_point_offsets의 값은 0 내지 (num_tile_columns_minu1 + 1)*(num_tile_rows_minus1 + 1) - 1의 범위 내에 존재하여야만 한다. tiles_or_entropy_coding_sync_idc가 2와 동일할 때, num_entry_point_offsets의 값은 0 내지 PicHeightInCtbs-1의 범위 내에 존재하여야만 한다. 존재하지 않을 때, num_entry_point_offsets의 값은 0과 동일한 것으로 추론된다.

[0180] **offset_len_minus1** + 1은 entry_point_offset[i] 구문 요소들의 비트들 내의 길이를 지정한다.

[0181] **entry_point_offset[i]**는 바이트들 내의 i번째 엔트리 지점 오프셋을 지정하며, offset_len_minu1 plus 1 비트들에 의해 표현되어야만 한다. 슬라이스 헤더 이후의 코딩된 슬라이스 데이터는 0부터 num_entry_point_offsets까지 범위의 지수 값을 갖는, num_entry_point_offsets + 1 서브셋들로 구성된다. 서브셋 0은 코딩된 슬라이스 데이터의 바이트 0 내지 entry_point_offset[0]-1로 구성되고, 1 내지 num_entry_point_offsets-1의 범위 내의 k를 갖는, 서브셋 k는 바이트들 entry_point_offset[k-1] 내지 entry_point_offset[k]+entry_point_offset[k-1]-1로 구성되며, 마지막 서브셋(num_entry_point_offsets와 동일한 서브셋 지수를 갖는)은 코딩된 슬라이스 데이터의 나머지 바이트들로 구성된다.

[0182] tiles_or_entropy_coding_sync_idc가 1과 동일하고 num_entry_point_offsets가 0보다 클 때, 각각의 서브셋은 정확하게 하나의 타일의 모든 코딩된 비트를 포함하여야만 하며, 서브셋들의 수(즉, num_entry_point_offsets + 1의 값)는 슬라이스 내의 타일들의 수와 동일하거나 또는 적어야만 한다.

[0183] 주의 - tiles_or_entropy_coding_sync_idc가 1과 동일할 때, 각각의 슬라이스는 반드시 하나의 타일(이 경우에 있어서 엔트로피 지점들의 시그널링이 불필요한)의 서브셋 또는 완전한 타일들의 정수를 포함하여야만 한다.

[0184] tiles_or_entropy_coding_sync_idc가 2와 동일하고 num_entry_point_offsets가 0보다 클 때, 0 내지 num_entry_point_offsets - 1의 범위 내의 k를 갖는 각각의 서브셋 k는 코딩 트리블록들이 정확하게 하나의 열의 모든 코딩된 비트를 포함해야만 하고, 마지막 서브셋(num_entry_point_offsets와 동일한 서브셋 지수를 갖는)은 슬라이스 내에 포함된 나머지 코딩 블록들의 모든 코딩된 비트를 포함해야만 하며, 나머지 코딩 블록들은 코딩 트리블록들의 정확하게 하나의 열 또는 코딩 트리블록들의 하나의 열의 하나의 서브셋으로 구성되며, 서브셋들의 수(즉, num_entry_point_offsets + 1의 값)는 슬라이스 내의 코딩 트리 블록들의 열들의 수와 동일해야만 하고, 슬라이스 내의 코딩 트리 블록들의 하나의 열의 서브셋이 또한 계수된다.

[0185] 주의 - tiles_or_entropy_coding_sync_idc가 2와 동일할 때, 슬라이스는 코딩 트리 블록들의 다수의 열 및 코딩 트리 블록들의 하나의 열의 하나의 서브셋을 포함할 수 있다. 예를 들면, 만일 하나의 슬라이스가 코딩 트리 블록들의 2½ 열을 포함하면, 서브셋들의 수(즉, num_entry_point_offsets + 1의 값)는 3과 동일해야만 한다.

[0186] 상응하는 화상 파라미터 세트 RBSP 구문은 도 22에 도시된 것과 같이 선택될 수 있다.

[0187] 화상 파라미터 세트 RBSP 시멘틱은 다음과 같을 수 있다:

[0188] 1과 동일한 **dependent_slice_enabled_flag**는 화상 파라미터 세트를 언급하는 코딩된 화상들을 위한 슬라이스 헤더 내의 구문 요소(dependent_slice_flag)의 존재를 지정한다. 0과 동일한 dependent_slice_enabled_flag는 화상 파라미터 세트를 언급하는 코딩된 화상들을 위한 슬라이스 헤더 내의 구문 요소(dependent_slice_flag)의 부재를 지정한다. tiles_or_entropy_coding_sync_idc가 3과 동일할 때, dependent_slice_enabled_flag의 값은 모두 1과 동일해야만 한다.

[0189] 0과 동일한 **tiles_or_entropy_coding_sync_idc**는 화상 파라미터 세트를 언급하는 각각의 화상 내에 단지 하나의 타일만이 존재하고 화상 파림터 세트를 언급하는 각각의 화상 내의 코딩 트리 블록들의 열의 첫 번째 코딩 블록을 디코딩하기 전에 적용되는 콘텍스트 변수들을 위한 어떠한 특정 동기화 과정이 존재해서는 안 되며, 화상 파라미터 세트를 언급하는 코딩된 화상들을 위한 cabac_independent_flag 및 dependent_slice_flag의 값은 모두 1과 동일해서는 안 된다는 것을 지정한다.

[0190] cabac_independent_flag 및 de[pendent_slice_flag가 슬라이스를 위하여 모두 1과 동일할 때, 슬라이스는 엔트로피 슬라이스인 것에 유의하여야 한다.

[0191] 1과 동일한 tiles_or_entropy_coding_sync_idc는 화상 파라미터 세트를 언급하는 각각의 화상 내에 하나 이상의 타일이 존재할 수 있는 것을 지정하고, 화상 파라미터 세트를 언급하는 각각의 화상 내의 코딩 트리 블록들의 하나의 열의 첫 번째 코딩 트리 블록을 디코딩하기 전에 적용되는 콘텍스트 변수들을 위한 어떠한 특정 동기화 과정도 존재해서는 안 되며, 화상 파라미터 세트를 언급하는 코딩된 화상들을 위한 cabac_independent_flag 및 dependent_slice_flag의 값들은 모두 1과 동일해서는 안 된다.

[0192] 2와 동일한 tiles_or_entropy_coding_sync_idc는 화상 파라미터 세트를 언급하는 각각의 화상 내에 하나의 타일만이 존재해야만 한다는 것을 지정하고, 화상 파라미터 세트를 언급하는 각각의 화상 내의 코딩 트리 블록들의 하나의 열의 첫 번째 코딩 트리 블록을 디코딩하기 전에 콘텍스트 변수들을 위한 특정 동기화 과정이 적용되어야만 하며, 화상 파라미터 세트를 언급하는 각각의 화상 내의 코딩 트리 블록들의 하나의 열의 두 개의 코딩 트리 블록을 디코딩한 후에 콘텍스트 변수들을 위한 특정 기억 과정이 적용되어야만 하며, 화상 파라미터 세트를 언급하는 코딩된 화상들을 위한 cabac_independent_flag 및 dependent_slice_flag의 값들은 모두 1과 동일해서는 안 된다.

[0193] 3과 동일한 tiles_or_entropy_coding_sync_idc는 화상 파라미터 세트를 언급하는 각각의 화상 내에 하나의 타일만이 존재해야만 한다는 것을 지정하고,

[0194] 화상 파라미터 세트를 언급하는 각각의 화상 내의 코딩 트리 블록들의 하나의 열의 첫 번째 코딩 트리 블록을 디코딩하기 전에 적용되는 콘텍스트 변수들을 위한 어떠한 특정 동기화 과정도 존재해서는 안 되며, 화상 파라미터 세트를 언급하는 코딩된 화상들을 위한 cabac_independent_flag 및 dependent_slice_flag의 값들은 모두 1과 동일할 수 있다.

[0195] dependent_slice_enabled_flag가 0과 동일해야만 할 때, tiles_or_entropy_coding_sync_idc는 3과 동일해서는 안 된다.

[0196] tiles_or_entropy_coding_sync_idc의 값이 코딩된 비디오 시퀀스 내에서 작동되는 모든 화상 파라미터 세트를 위하여 동일해야만 하는 것이 비트스트림 적합성의 요구조건이다.

[0197] 화상 파라미터 세트를 언급하는 각각의 슬라이스를 위하여, tiles_or_entropy_coding_sync_idc가 2와 동일하고 슬라이스 내의 첫 번째 코딩 블록이 코딩 트리 블록들의 하나의 열의 첫 번째 코딩 트리 블록 내의 첫 번째 코딩 블록이 아닐 때, 글라이스 내의 마지막 코딩 블록은 슬라이스 내의 첫 번째 코딩 블록과 동일한 코딩 트리 블록들의 열에 속해야만 한다.

[0198] **num_tile_column_minus1** + 1은 화상을 분할하는 타일 행들의 수를 지정한다.

[0199] **num_tile_rows_minus1** + 1은 화상을 분할하는 타일 열들의 수를 지정한다.

[0200] num_tile_column_minus1이 0과 동일할 때, num_tile_rows_minus1은 0과 동일해서는 안 된다. 1과 동일한 uniform_spacing_flag는 행 경계들과 유사한 열 경계들이 화상을 가로질러 균일하게 분포되는 것을 지정한다. 0과 동일한 uniform_spacing_flag는 행 경계들과 유사한 열 경계들이 화상을 가로질러 균일하게 분포되지 않으나, 구문 요소들 column_width[i] 및 row_height[i]를 사용하여 분명하게 시그널링되는 것을 지정한다.

[0201] **column_width**[i]는 코딩 트리블록들의 유닛들 내의 i번째 타일 행의 폭을 지정한다.

[0202] **row_height**[i]는 코딩 트리블록들의 유닛들 내의 i번째 타일 열의 높이를 지정한다.

[0203] 벡터 colWidth[i]는 0부터 num_tile_columns_minus1까지의 범위의 행(i)을 갖는 CTB들의 유닛들 내의 i번째 타일의 폭을 지정한다.

[0204] 벡터 CtbAddrRStoTS[ctbAddrRS]는 래스터 스캔 순서 내의 CTB 주소로부터 0부터 (picHeightInCtbs*picWidthInCtbs) - 1까지 범위의 지수(ctbAddrRS)를 갖는 타일 스캔 순서 내의 CTB 주소까지의 대화를 지정한다.

[0205] 벡터 CtbAddrRStoTS[ctbAddrTS]는 래스터 스캔 순서 내의 CTB 주소로부터 0부터 (picHeightInCtbs*picWidthInCtbs) - 1까지 범위의 지수(ctbAddrTS)를 갖는 타일 스캔 순서 내의 CTB 주소까지의 대화를 지정한다.

[0206] 벡터 TileId[ctbAddrTS]는 0부터 (picHeightInCtbs*picWidthInCtbs) - 1까지 범위의 지수(ctbAddrTS)를 갖는 타일 스캔 순서 내의 CTB 주소로부터 타일 id까지의 대화를 지정한다.

[0207] colWidth, CtbAddrRStoTS, CtbAddrTStoRS 및 TileId의 값들은 입력들로서 PicHeightInCtbs 및 PicWidthInCtbs를 갖는 CTB 래스터 및 타일 스캐닝 대화 과정을 적용시킴으로써 유래되고, 출력은 colWidth, CtbAddrRStoTS 및 TileId에 지정된다.

[0208] 루마 샘플들 내의 유닛들의 i번째 타일 행의 폭을 지정하는 ColumnWidthInLumaSamples[i]의 값들은 colWidth[i]≪Log2CtbSize와 동일하게 설정된다.

[0209] 0 내지 picWidthInMinCbs -1까지의 범위를 갖는 x와 0 내지 picHeightInMinCbs -1까지의 범위를 갖는 y를 갖는, 최소 코딩 블록들의 유닛들 내의 위치(x, y)로부터 z-스캔 순서 내의 최소 코딩 블록 주소까지의 대화를 지정하는, 어레이 MinCbAddrZS[x][y]는 입력으로서 Log2MinCbSize, Log2CtbSize, PicHeightInCtbs, PicWidthInCtbs, 및 벡터 CtbAddrRStoTS를 갖는 Z 스캐닝 순서 어레이 초기화 과정을 적용시킴으로써 유래되고, 출력은 MinCbAddrZS에 지정된다.

[0210] 1과 동일한 **loop_filter_across_tiles_enabled_flag**는 타일 경계들을 가로질러 인-루프(in-loop) 필터링 연산들이 실행되는 것을 지정한다. 0과 동일한 loop_filter_across_tiles_enabled_flag는 타일 경계들을 가로질러 인-루프 필터링 연산들이 실행되지 않는 것을 지정한다. 인-루프 필터링 연산들은 블록화 제거 필터, 샘플 적응적 오프셋, 및 적응적 루프 필터 연산들을 포함한다. 존재하지 않을 때, loop_filter_across_tiles_enabled_flag의 값은 1과 동일한 것으로 추론된다.

[0211] 0과 동일한 **cabac_independent_flag**는 슬라이스 내의 코딩 블록들의 CABAC 디코딩이 이전에 디코딩된 슬라이스의 어떠한 상태와도 독립적인 것을 지정한다. 1과 동일한 cabac_independent_flag는 슬라이스 내의 코딩 블록들의 CABAC 디코딩이 이전에 디코딩된 슬라이스의 어떠한 상태에 의존한다는 것을 지정한다. 존재하지 않을 때, cabac_independent_flag의 값은 0과 동일한 것으로 추론된다.

[0212] 최소 코딩 블록 주소를 갖는 코딩 블록의 이용가능성을 위한 도출 과정은 다음과 같을 수 있다:

[0213] 이러한 과정에 대한 입력들은 다음과 같다:

[0214] - z-스캔 순서 내의 최소 코딩 블록 주소 minCbAddrZS

[0215] - z-스캔 순서 내의 현재 최소 코딩 블록 주소 currminCBAddrZS

[0216] 이러한 과정의 출력은 z-스캔 순서(cbAvailable) 내의 최소 코딩 블록 주소(cbAddrZS)를 갖는 코딩 블록의 이용가능성이다.

[0217] 주의 1 - 이용가능성의 의미는 이러한 과정이 적용될 때 결정된다.

[0218] 주의 2 - 그것의 관계없이, 어떠한 코딩 블록이 z-스캔 순서 내의 최소 코딩 블록 크기를 갖는 코딩 블록의 주소인, 최소 코딩 블록 주소와 관련된다.

[0219] - 만일 다음의 조건들 중 하나 또는 그 이상이 사실이면, cbAvailable은 거짓(FALSE)으로 설정된다.

[0220] - minCbAddrZS는 0보다 작다.

[0221] - minCbAddrZS는 currMinCBAddrZS보다 크다.

[0222] - 최소 코딩 블록 주소(minCbAddrZS)를 갖는 코딩 블록은 현재 최소 코딩 블록(currMinCBAddrZS)을 갖는 코딩 블록과 다른 주소에 속하며 현재 최소 코딩 블록(currMinCBAddrZS)을 갖는 코딩 블록을 포함하는 슬라이스의 dependent_slice_flag는 0과 동일하다.

[0223] - 그렇지 않으면, cbAvailable은 사실(TRUE)로 설정된다.

[0224] 슬라이스 데이터를 위한 CABAC 파싱 과정은 다음과 같을 수 있다:

[0225] 이러한 과정은 기술자 ae(v)를 갖는 특정 구문 요소들을 파싱할 때 적용된다.

[0226] 이러한 과정에 대한 입력들은 구문 요소의 값 및 이전에 파싱된 구문 요소들의 값들의 요구이다.

[0227] 이러한 과정의 출력은 구문 요소의 값이다.

[0228] 슬라이스의 슬라이스 데이터의 파싱을 시작할 때, CABAC 파싱 과정의 초기화 과정이 적용된다.

[0229] 도 23은 현재 코딩 트리 블록(유용한)에 대하여 코딩 트리 블록 이용가능성 도출 과정을 적용하기 위하여 공간적 이웃 블록(T)이 어떻게 사용되는지에 대하여 도시한다.

[0230] 공간적 이웃 블록(T, 도 23)을 포함하는 코딩 트리 블록의 최소 코딩 블록 주소, ctbMINCbAddrT는 다음과 같이 현재 코딩 트리 블록이 상부-왼쪽 루마 샘플의 위치(x0, y0)를 사용하여 도출된다.

[0231] x= x0 + 2 ≪ Log2CtbSize - 1

[0232] y = y0 - 1

[0233] ctbMinCbAddrT = MinCbAddrZS[x ≫ Log2MinCbSize][y ≫ Log2MinCbSize]

[0234] 변수(availableFlagT)는 입력으로서 ctbMinCbAddrT를 갖는 코딩 블록 이용가능성을 적용시킴으로써 획득된다.

[0235] 산술 디코딩 엔진은 다음과 같이 초기화된다.

[0236] 만일 CtbAddrRS가 slice_address와 동일하면, dependent_slice_flag는 1과 동일하고 wntropy_config_reser_flag는 0과 동일하며, 다음이 적용된다.

[0237] CABAC 파싱 과정의 동기화 과정은 입력으로서 TableStateIdxDS 및 TableMPSValDS와 함께 적용된다.

[0238] 종료 이전에 이진 결정들을 위한 디코딩 과정이 적용되고, 산술 디코딩 엔진을 위한 초기화 과정이 뒤따른다.

[0239] 그렇지 않고 만일 tiles_or_entropy_coding_sync_idc가 2와 동일하고, CtbAddrRS%PicWidthInCtbs가 0과 동일하면, 다음이 적용된다.

[0240] availableFlagT가 1과 동일할 때, CABAC 파싱 과정을 위한 동기화 과정은 입력으로서 TableStateIdxWPP 및 TableMPSValWPP와 함께 적용된다.

[0241] 종료 이전에 이진 결정들을 위한 디코딩 과정이 적용되고, 산술 디코딩 엔진을 위한 초기화 과정이 뒤따른다.

[0242] cabac_indenpendent_flag가 0과 동일하고 dependent_slice_flag가 1과 동일할 때, 또는 tiles_or_entropy_coding_sync_idc가 2와 동일할 때, 기억 과정은 다음과 같이 적용된다.

[0243] tiles_or_entropy_coding_sync_idc가 2와 동일하고 CtbAddrRS%PicWidthInCtbs가 2와 동일할 때, CABAC 파싱 과정의 기억 과정은 출력으로서 TableStateIdxWPP 및 TableMPSValWPP와 함께 적용된다.

[0244] cabac_independent_flaf가 0과 동일하고, dependent_slice_flag가 1과 동일하며, 둥_of_slice_flag가 1과 동일하면, CABAC 파싱 과정의 기억 과정은 출력으로서 TableStateIdxDS 및 TableMPSValDS와 함께 적용된다.

[0245] 구문 요소들의 파싱은 다음과 같이 진행된다:

[0246] 각각의 구문 요소의 요구된 값을 위하여 이진화가 도출된다.

[0247] 구문 요소를 위한 이진화 및 파싱된 빈(bin)들의 시퀀스는 디코딩 과정 흐름을 결정한다.

[0248] 변수(binIdx)에 의해 인덱싱되는, 구문 요소의 이진화의 각각의 빈을 위하여, 콘텍스트 지수(ctxIdx)가 도출된다.

[0249] 각각의 ctxIdx를 위하여, 산술 디코딩 과정이 적용된다.

[0250] 파싱된 빈들의 결과로서 생기는 시퀀스(b0..bbinIdx)는 각각의 빈의 디코딩 후에 이진화 과정에 의해 주어지는 빈 스트링(bin string)들의 세트와 비교된다. 시퀀스가 주어진 세트 내의 빈 스트링과 일치할 때, 상응하는 값이 구문 요소에 지정된다.

[0251] 구문 요소의 값을 위한 요구가 구문 요소(pcm_flag)를 위하여 진행되고 pcm_flag의 디코딩된 값이 1과 동일한 경우에, 디코딩 엔진은 어떠한 pcm_alighment_zero_bit, num_subsequent_pcm, 및 모든 pcm_sample_luma와 pcm_sample_chroma 데이터의 디코딩 후에 초기화된다.

[0252] 따라서, 위의 설명은 도 24에 도시된 것과 같은 디코더를 나타낸다. 일반적으로 도면부호 5에 의해 표시되는, 이러한 디코더는 화상(10)이 분할되는 슬라이스들(14)의 유닛들 내에 화상(10)이 코딩되는 데이터 스트림(12)으로부터 화상(10)을 재구성하며, 디코더(5)는 슬라이스 순서(16)에 따라 데이터 스트림(12)으로부터 슬라이스들(14)을 디코딩하도록 구성된다. 자연적으로, 디코더(5)는 슬라이스들(14)을 연속으로 디코딩하도록 제한되지 않

는다. 오히려, 디코더(5)는 슬라이스들(14) 내로 분할하는 화상(10)이 파면 병렬 처리에 적합하면, 슬라이스들(14)을 디코딩하기 위하여 파면 병렬 처리를 사용할 수 있다. 따라서, 디코더(5)는 예를 들면, 위에 설명되고 또한 아래에 설명될 것과 같이 파면 처리를 허용하기 위하여 슬라이스 순서(16)를 고려함으로써 슬라이스들(14)의 디코딩의 시작을 갖는 스태그드 방식(staggered manner)으로 병렬로 슬라이스들(14)을 디코딩할 수 있는 디코더일 수 있다.

[0253] 디코더(5)는 적어도 두 가지 방식(20 및 22) 중 하나에 따라 현재 슬라이스를 디코딩하기 위하여 슬라이스들(14)의 현재 슬라이스 내의 구문 요소 부(18)에 응답한다. 적어도 두 가지 방식 중 첫 번째 방식, 즉 방식(20)에 따라, 현재 슬라이스는 슬라이스 경계들을 가로질러, 즉 도 24의 쇄선을 가로질러 콘텍스트의 도출을 포함하는 콘텍스트 적응적 엔트로피 디코딩(context-adaptive entropy coding)을 사용하여, 즉 다른 "슬라이스들을 선행하는 슬라이스 순서(16)로" 코딩/디코딩에서 기인하는 정보를 사용함으로써 데이터 스트림(12)으로부터 디코딩된다. 또한, 첫 번째 방식(20)을 사용하는 데이터 스트림(12)으로부터의 현재 슬라이스의 디코딩은 이전에 디코딩된 슬라이스의 심볼 확률의 저장된 상태들에 의존하는, 현재 슬라이스의 디코딩의 시작에서 코덱의 심볼 확률들의 연속적인 업데이트 및 심볼 확률들의 초기화를 포함한다. 그러한 의존성은 예를 들면 "코덱 변수들을 위한 동기화 과정"과 함께 설명되었다. 최종적으로 첫 번째 방식(20)은 또한 슬라이스 경계들을 가로질러 예측 디코딩을 포함한다. 슬라이스 경계들을 가로지르는 그러한 예측 디코딩은 예를 들면, 슬라이스 경계들을 가로지르는 인트라-예측, 즉 슬라이스를 선행하는, "슬라이스 순서(16)"로 이미 재구성된 샘플 값들을 기초로 하여 현재 슬라이스 내의 샘플 값들의 예측, 혹은 움직임 벡터들, 예측 방식들, 코딩 방식들 등의 예측과 같은 슬라이스 경계들을 가로지르는 코딩 파라미터들의 예측을 포함할 수 있다.

[0254] 두 번째 방식(22)에 따르면, 디코더(5)는 슬라이스 경계들을 가로지르지 않기 위하여 콘텍스트들의 도출의 제한을 갖는, 콘텍스트 적응적 엔트로피 디코딩을 사용하여, 데이터 스트림(12)으로부터 현재 슬라이스, 즉 현재 디코딩되는 슬라이스를 디코딩한다. 만일 그렇더라도, 예를 들면, 현재 슬라이스 내의 블록과 관련하여 특정 구문 요소를 위한 콘텍스트를 도출하기 위하여 사용되는 이웃하는 위치들의 템플릿(template)은 이웃하는 슬라이스 내로 확장하며, 그렇게 함으로써 현재 슬라이스의 슬라이스 경계를 가로지르며, 이웃하는 슬라이스의 이러한 이웃하는 부의 상응하는 구문 요소의 값과 같은, 이웃하는 슬라이스의 각각의 부분의 상응하는 속성은 현재 슬라이스와 이웃하는 슬라이스들 사이의 상호의존을 억제하기 위하여 디폴드 값(default value)으로 설정된다. 마치 첫 번째 방식(20)의 경우처럼 콘텍스트들의 심볼 확률들의 연속적인 업데이트가 발생할 수 있으나, 두 번째 방식(20)에서의 심볼 확률들의 초기화는 어떠한 이전에 디코딩된 슬라이스와는 독립적이다. 또한, 슬라이스 경계들을 가로지르지 않기 위하여 예측 디코딩의 한정을 갖는 예측 디코딩이 실행된다.

[0255] 도 24의 설명 및 그 다음의 설명의 이해를 용이하게 하기 위하여, 도 24와 비교하여 더 구체적인 의미에서의 디코더(5)의 가능한 구현을 도시하는, 도 25가 참조된다. 도 24의 경우에서와 같이, 디코더(5)는 예를 들면, 예측 잔류 및 예측 파라미터들을 획득하기 위하여 데이터 스트림을 디코딩하기 위한 콘텍스트 적응적 엔트로피 디코딩을 사용하는 예측 디코더이다.

[0256] 도 25에 도시된 것과 같이, 디코더(5)는 엔트로피 디코더(24), 탈양자화 및 역 변환 모듈(26), 도 25에 도시된 것과 같이 예를 들면, 가산기 및 예측기(28)로서 구현되는 결합기(28)를 포함할 수 있다. 엔트로피 디코더(24), 모듈(26) 및 가산기(27)는 언급된 순서로 디코더(5)의 입력과 출력 사이에 순차적으로 연결되고, 예측기(28)는 결합기(27)와 함께 예측 루프를 형성하기 위하여 가산기(28)의 출력 및 그것들의 또 다른 입력 사이에 연결된다. 따라서, 디코더(24)는 예측기(28)의 코딩 파라미터 입력에 부가적으로 연결되는 출력을 갖는다.

[0257] 도 25가 디코더가 현재 화상을 순차적으로 디코딩하는 인상을 제공하나, 디코더(25)는 예를 들면, 병렬로 화상(10)을 디코딩하도록 구현될 수 있다. 디코더는 예를 들면, 각각 도 25의 요소들(24-28)에 따라 작동하는 다수의 코어를 포함할 수 있다. 그러나, 병렬 처리는 선택적이며, 엔트로피 디코더(24)의 입력에서 오는 데이터 스트림을 디코딩하기 위하여 순차적으로 작동하는 디코더(5)가 또한 가능할 수 있다.

[0258] 현재 화상(10)을 순차적으로 또는 병렬로 디코딩하는 방금 언급된 능력을 효율적으로 달성하기 위하여, 디코더(5)는 화상(10)을 디코딩하기 위하여 코딩 블록들(30)이 유닛들 내에 작동한다. 코딩 블록들(30)은 예를 들면, 코딩 트리 블록들 또는 가장 큰 코딩 블록들(32)이 쿼드트리 분할과 같은 재귀적 분할에 의해 분할되는 리프 블록(leaf block)들이다. 코드 트리블록들(32)은 차례로, 화상(10)의 이러한 코드 트리블록들(32)로의 규칙적인 분할을 형성하기 위하여 행들과 열들 내에 규칙적으로 배치된다. 도 25에서, 코드 트리블록들(32)은 실선으로 도시되고, 코딩 블록들(30)은 쇄선으로 도시된다. 설명의 목적을 위하여, 하나의 코드 트리블록(32)만이 코딩 블록들(30)로 더 분할되는 것으로 도시되나, 대신에, 나머지 코드 트리블록들(32)은 직접적으로 코딩 블록을 형

성하기 위하여 더 분할되지 않도록 도시된다. 데이터 스트림(12)은 화상(10)이 코드 블록들(30)로 어떻게 분할되는지에 대하여 시그널링하는 구문 부를 포함할 수 있다.

[0259] 데이터 스트림(12)은 각각의 코딩 블록(30)을 위하여, 모듈들(24 내지 28)이 어떻게 그러한 코딩 블록(30) 내의 화상 콘텐츠를 복원하는지를 나타내는 구문 요소들을 전달한다. 예를 들면, 이러한 구문 요소들은 다음을 포함한다:

[0260] 1) 선택적으로, 코딩 블록(30)을 예측 블록들로 더 분할하는 분할 데이터,

[0261] 2) 선택적으로, 코딩 블록(30)을 잔류(residual) 및/또는 변환 블록들로 더 분할하는 분할 데이터,

[0262] 3) 코딩 블록(30)을 위한 예측 신호를 도출하기 위하여 어떠한 예측 방식이 사용되는지를 시그널링하는 예측 방식, 이러한 예측 방식이 시그널링되는 입상도(granularity)는 코딩 블록들(30) 및/또는 예측 블록에 의존할 수 있다.

[0263] 4) 예측 파라미터들이 코딩 블록 당 또는 만일 존재하면, 예를 들면 예측 방식에 따라 보내지는 일종의 예측 파라미터들을 갖는 예측 블록 당 시그널링될 수 있다. 가능한 예측 방식들은 예를 들면, 인트라-예측 및/또는 인터-예측을 포함할 수 있다.

[0264] 5) 예측 신호 및/또는 재생산되도록 재구성되는 신호를 획득하기 위하여 화상(10)을 필터링하기 위한 필터링 정보와 같은 다른 구문 요소들이 또한 존재할 수 있다.

[0265] 6) 마지막으로, 예를 들면, 그중에서도 변환 계수들의 형태의 잔류 정보가 코딩 블록(30)을 위한 데이터 스트림 내에 포함될 수 있으며; 잔류 블록들의 유닛들 내에, 잔류 데이터가 시그널링될 수 있으며; 잔류 블록 당, 스펙트럼 분해는 만일 존재하면, 앞서 언급된 변환 블록들의 유닛들 내에서 실행될 수 있다.

[0266] 엔트로피 디코더(24)는 데이터 스트림으로부터 앞서 언급된 구문 요소들을 획득하는데 책임이 있다. 이를 위하여, 엔트로피 디코더(24)는 콘텍스트 적응적 엔트로피 디코딩을 사용한다. 즉, 엔트로피 디코더(24)는 일부 콘텍스트를 제공한다. 데이터 스트림(12)으로부터 특정 구문 요소를 도출하기 위하여, 엔트로피 디코더(24)는 가능한 콘텍스트 중에서 특정 콘텍스트를 선택한다. 가능한 콘텍스트들 중에서의 선택은 현재 구문 요소가 속하는 화상(10) 부분의 이웃(neighborhood)의 속성에 의존한다. 각각의 가능한 콘텍스트들을 위하여, 엔트로피 디코더(24)는 심볼 확률들, 즉 어떠한 엔트로피 디코더(24)가 작동하는지를 기초로 하는 심볼 알파벳의 각각의 가능한 심볼을 위한 확률 추정을 관리한다. "관리"는 각각의 콘텍스트와 관련된 심볼 확률들을 실제 화상 콘텐츠에 적용하기 위하여 앞서 언급된 콘텍스트들의 심볼 확률들의 연속적인 업데이트를 포함한다. 이러한 측정에 의해, 심볼 확률들은 심볼들의 실제 확률 통계에 적용된다.

[0267] 이웃의 속성들이 현재 코딩 블록(30)과 같은 화상(10)의 현재 부분에 영향을 미치는 또 다른 상황은 예측기(28) 내의 예측 디코딩이다. 예측은 현재 코딩 블록(30) 내의 예측 콘텐츠에만 한정되는 것이 아니라, 또한 예측 파라미터들, 분할 데이터, 또는 심지어 변환 계수들과 같은 현재 코딩 블록(30)을 위한 데이터 스트림 내에 포함되는 파라미터들의 예측도 포함할 수 있다. 즉, 예측기(28)는 데이터 스트림(12)으로부터 모듈(26)에 의해 획득된 것과 같이 예측 잔류와 결합되는 기록된 신호를 획득하기 위하여 화상 콘텐츠 또는 앞서 언급된 이웃으로부터의 그러한 파라미터들을 예측한다. 예측 파라미터들의 경우에 있어서, 예측기(28)는 예측 파라미터의 실제 값을 획득하기 위하여 예측 잔류들로서 데이터 스트림 내에 포함된 구문 요소들을 사용할 수 있다. 예측기(28)는 결합기(27)에서 예측 잔류와 결합되는 앞서 언급된 예측 신호를 획득하기 위하여 후자의 예측 파라미터 값을 사용한다.

[0268] 앞서 언급된 "이웃"은 주로 현재 엔트로피 디코딩되는 구문 요소 또는 현재 예측 디코딩되는 구문 요소가 속하는 현재 부분의 원주의 상부 왼쪽 부분을 포함한다. 도 25에서, 각각의 이웃이 바람직하게는 하나의 코딩 블록(30)을 위하여 34에 도시된다.

[0269] 코딩/디코딩 순서는 코딩 블록들(30) 중에서 정의되며, 가장 거친 레벨에서, 화상(10)의 코드 트리블록들(32)은 여기서는 상부로부터 하부로의 열 방식(row-wise)에 이르게 하는 래스터 스캔으로서 도시되는, 스캔 순서(36) 내에서 스캔된다. 각각의 코드 트리블록 내에, 코딩 블록들(30)은 각각의 계층 내에서, 코드 트리블록(32)이 실질적으로 또한 상부로부터 하부로의 열 방식에 이르게 하는 래스터 스캔 내에서 스캔되는 것과 같이 첫 번째 깊이 순회 순서(depth first traversal order)로 스캔된다.

[0270] 코딩 블록들(30) 중에서 정의되는 코딩 순서는 이웃(34)이 이미 코딩 순서에 따라 디코딩의 대상이 된 화상(10)의 부들을 대부분 포함한다는 점에서 콘텍스트들을 선택하거나 및/또는 공간 예측을 실행하기 위하여 이웃 내

의 속성을 도출하도록 사용되는 이웃(34)의 정의와 조화를 이룬다 이웃(34)의 부들이 화상(10)의 비-이용가능한 부들을 포함할 때마다, 예를 들면, 디폴트 데이터가 대신 사용된다. 예를 들면, 이웃 템플릿(34)은 화상(10) 외부로 확장할 수 있다. 그러나, 또 다른 가능성은 이웃(34)이 이웃하는 슬라이스 내로 확장하는 것이다.

[0271] 슬라이스들은 예를 들면, 코딩 블록들(30)을 따라 정의되는 코딩/디코딩 순서를 따라 화상(120)을 세분하는데, 즉, 각각의 슬라이스는 앞서 언급된 코딩 블록 순서를 따라 코딩 블록들(30)의 연속적인 비-차단된 시퀀스이다. 도 25에서, 슬라이스들은 일점쇄선(14)으로 표시된다. 슬라이스들 중에서 정의되는 순서는 위에 설명된 것과 같이 순차적 코딩 블록들(30)의 런(run)들의 구성으로부터 야기한다. 만일 특정 슬라이스(14)의 구문 요소 부(18)가 첫 번째 방식으로 동일하게 디코딩되는 것을 나타내면, 엔트로피 디코더(24)는 슬라이스 경계들을 가로질러 콘텍스트들을 도출하기 위하여 콘텍스트 적응적 엔트로피 디코딩을 허용한다. 즉, 현재 슬라이스(14)와 관련하여 엔트로피 디코딩 데이터 내의 콘텍스트들을 선택하기 위하여 공간적 이웃(34)이 사용된다. 도 25의 경우에 있어서, 예를 들면, 슬라이스 수(3)는 현재 디코딩되는 슬라이스일 수 있으며, 코딩 블록(30) 또는 그 안에 포함된 일부 부에 대한 엔트로피 디코딩 구문 요소에 있어서, 엔트로피 디코더(24)는 슬라이스 수(1)와 같은 이웃하는 슬라이스 내의 부들의 디코딩으로부터 기인하는 속성들을 사용할 수 있다. 예측기(28)는 동일하게 행동하며, 첫 번째 방식(20)인 슬라이스를 위하여, 예측기(28)는 현재 슬라이스를 둘러싸는 슬라이스 경계를 가로질러 공간 예측을 사용한다.

[0272] 그러나, 그것들과 관련된 두 번째 방식(22)을 갖는, 즉 구문 요소 부(18)가 두 번째 방식(22)을 나타내는 슬라이스들을 위하여, 엔트로피 디코더(24) 및 예측기(28)는 현재 슬라이스 내에만 있는 부들에 대한 속성들에 의존하기 위하여 엔트로피 콘텍스트들의 도출과 예측 디코딩을 제한한다. 명백하게, 코딩 효율은 이러한 제한으로부터 어려움을 겪는다. 다른 한편으로, 두 번째 방식(22)의 슬라이스들은 슬라이스들의 시퀀스 사이의 상호의존성의 방해를 허용한다. 따라서, 두 번째 방식(22)의 슬라이스들은 재동기화 지점들을 허용하기 위하여 화상(10) 내에 또는 화상(10)이 속하는 비디오 내에 배치될 수 있다. 그러나 각각의 화상(10)이 두 번째 방식(22)에서 적어도 하나의 슬라이스를 가질 필요는 없다.

[0273] 위에 이미 언급된 것과 같이, 첫 번째 및 두 번째 방식(20 및 22)은 또한 심볼 확률들의 그것들의 초기화가 다르다. 두 번째 방식(22)으로 코딩된 슬라이스들은 어떠한 이전에 디코딩된, 즉 슬라이스들 중에서 정의된 순서의 의미로 이전에 디코딩된 슬라이스와 관계없이 확률들을 재초기화하는 엔트로피 디코더(24)를 야기한다. 심볼 확률들은 예를 들면, 인코더 면 및 디코더 면 모두에 알려진 디폴트 값들로 설정되거나, 또는 초기화 값들은 두 번째 방식(22)으로 코딩된 슬라이스들 내에 포함된다.

[0274] 즉, 두 번째 방식(22)으로 코딩되는/디코딩되는 슬라이스들을 위하여, 심볼 확률들의 적응은 항상 이러한 슬라이스들의 시작으로부터 즉시 시작한다. 따라서, 적응 정확도는 이러한 슬라이스들의 시작에서 이러한 슬라이스들을 위하여 좋지 않다.

[0275] 첫 번째 방식으로 코딩된/디코딩된 슬라이스들에서는 다르다. 후자의 슬라이스들을 위하여, 엔트리 디코더(24)에 의해 실행되는 심볼 확률들의 초기화는 이전에 디코딩된 슬라이스의 심볼 확률들의 저장된 상태들에 의존한다. 첫 번째 방식(20)에서 코딩된/디코딩된 슬라이스가 예를 들면, 화상(10)의 왼쪽 면 이외에, 즉 하부 쪽으로 그 다음의 열로 진행하기 전에 래스터 스캔(36)이 열 방식으로의 구동을 시작하는 면이 아닌 곳에 위치되는, 그것의 시작을 가질 때마다. 바로 선행하는 슬라이스의 엔트로피 디코딩의 끝에서 야기하는 갓과 같은 심볼 확률들이 적용된다. 이는 예를 들면 슬라이스 #4를 위한 화살표(38)에 의해 도 2에 도시된다. 슬라이스 #4는 화상(10)의 오른쪽 면과 왼쪽 면 사이의 어딘가에서 그것의 시작을 가지며, 따라서 심볼 확률들을 초기화하는데 있어서, 엔트로피 디코더(24)는 그것들의 끝까지, 즉 그것의 끝까지 슬라이스 #3의 엔트로피 디코딩 동안에 심볼 확률들의 연속적인 업데이트를 포함하여, 바로 선행하는 슬라이스, 즉 슬라이스 #3의 엔트로피 디코딩에서 획득된 심볼 확률들을 적용한다.

[0276] 그러나 예를 들면 슬라이스 #5와 같은 화상(10)의 왼쪽 면에서의 시작을 갖는, 그것과 관련된 두 번째 방식(22)을 갖는 슬라이스들은 바로 선행하는 슬라이스 #4의 엔트로피 디코딩의 종료 후에 획득되는 것과 같은 심볼 확률들을 적용하지 않는데, 그 이유는 두 번째 방식은 디코더(5)가 파면 처리의 사용에 의한 화상(10)의 병렬 디코딩을 방지하기 때문이다. 오히려, 위에 설명된 것과 같이, 엔트로피 디코더(24)는 화살표(40)에 의해 도시된 것과 같이 바로 선행하는(인코딩/디코딩 순서(36) 내의) 코드 트리블록 열 내의 두 번째(인코딩/디코딩 순서(36) 내의) 코드 블록(32)의 엔트로피 디코딩의 종료 후에 획득된 것과 같은 심볼 확률들을 적용한다.

[0277] 도 25에서, 예를 들면 화상(10)은 바람직하게는 코드 트리블록들의 3개의 열 및 코딩 트리 루트 블록들(32)의 4개의 행으로 분할되었고 각각의 코드 트리블록 열은 2개의 슬라이스로 세분화되었으며, 따라서 매 두 번째 슬라

이스의 시작은 각각의 코드 트리블록 열의 코딩 유닛 순서 내의 첫 번째 코딩 유닛과 일치한다. 따라서, 엔트로피 디코더(24)는 첫 번째 또는 맨 위의 코드 트리 루트 블록 열로 시작하고 그리고 나서 두 번째 및 그리고 나서 세 번째로 시작하는 스태그드 방식으로 이러한 코드 트리 루트 블록 열들의 디코딩의 시작과 함께, 각각의 코드 트리 루트 블록 열을 병렬로 디코딩함으로써, 화상(10)을 디코딩하는데 파면 처리를 사용할 수 있다.

[0278] 자연적으로, 재귀적 방식으로 또 다른 코딩 블록들(30)로의 블록들(32)의 분할은 선택적이며, 따라서 더 일반적인 의미에서, 블록들(32)은 또한 "코딩 블록들"로 불릴 수 있다. 즉, 더 일반적으로, 화상(10)은 열들과 행들로 배치되고 서로 중에서 정의되는 래스터 스캔 순서를 갖는 코딩 블록들로 분할될 수 있으며, 디코더(5)는 서브셋들이 슬라이스 순서에 따른 래스터 스캔 순서(36)를 따라 서로 뒤따르도록 하기 위하여 각각의 슬라이스를 래스터 스캔 순서(36) 내의 코딩 블록들(32)의 연속적인 서브셋과 연관시키도록 고려될 수 있다.

[0279] 또한 위의 설명으로부터 자명한 것과 같이, 디코더(5) 또는 더 구체적으로 엔트로피 디코더(24)는 래스터 스캔 순서(36)에 따라 코딩 블록 열의 두 번째 코딩 블록까지 어떠한 슬라이스의 콘텍스트 적응적 엔트로피 디코딩에서 획득되는 것과 같은 심볼 확률들을 저장하도록 구성될 수 있다. 그것과 관련된 첫 번째 방식(20)을 갖는 현재 슬라이스의 콘텍스트 적응적 엔트로피 디코딩을 위한 심볼 확률들의 초기화에 있어서, 디코더(5) 또는 , 더 구체적으로 엔트로피 디코더(24)는 현재 슬라이스와 관련된 코딩 블록들(32)의 연속적인 서브셋들의 첫 번째 코딩 블록(32)이 래스터 스캔 순서에 따른 열의 첫 번째 코딩 블록인지에 대하여 검사하도록 구성될 수 있다. 만일 그렇다면, 현재 슬라이스의 콘텍스트 적응적 엔트로피 디코딩을 위한 심볼 확률들은 화살표(40)와 관련하여 설명되는 것과 같이, 즉 래스터 스캔 순서(36)에 따라 열의 두 번째 코딩 블록까지 이전에 디코딩된 슬라이스의 콘텍스트 적응적 엔트로피 디코딩에서 획득되는 것과 같은 저장된 심볼 확률들에 따라 초기화된다. 만일 그렇지 않으면, 현재 슬라이스의 콘텍스트 적응적 엔트로피 디코딩을 위한 심볼 확률들의 초기화는 이전에 디코딩된 슬라이스의 끝까지 이전에 디코딩된 슬라이스의 콘텍스트 적응적 엔트로피 디코딩에서 획득되는 것과 같은 심볼 확률들에 따라 , 즉 화살표(38)에 따라 실행된다. 다시, 38에 따른 초기화의 경우에 있어서, 이는 슬라이스 순서(36) 내의 바로 선행하는 슬라이스의 엔트로피 디코딩의 끝에서의 저장된 상태로 여겨지며, 반면에, 초기화(40)의 경우에 있어서, 이는 블록 순서(36) 내의 블록(32)의 바로 선행하는 열의 두 번째 블록의 끝을 포함하는 이전에 디코딩된 슬라이스이다..

[0280] 도 24의 쇄선에 의해 도시된 것과 같이, 디코더는 적어도 세 가지 방식 중 어느 하나에 따라 현재 슬라이스를 디코딩하기 위하여 슬라이스들의 현재 슬라이스 내의 구문 요소 부(18)에 응답하도록 구성될 수 있다. 즉, 두 가지 방식(20 및 22) 이외에 세 번째 방식(42)이 존재할 수 있다. 세 번째 방식(42)은 슬라이스 경계들을 가로지르는 예측이 허용된다는 점에서 두 번째 방식(22)과 다를 수 있으나, 슬라이스 경계들을 가로지르지 않도록 하기 위하여 엔트로피 코딩/디코딩은 여전히 제한된다.

[0281] 위에서 구문 요소 부(18)와 관련하여 두 가지 실시 예가 설명되었다. 아래의 테이블은 이러한 두 가지 실시 예를 요약한다.

[0282]

	실시 예 1	실시 예 2
구문 요소 부	dependent_slice_flag, no_cabac_reset_flag	dependent_slice_flag
방식 1	dependent_slice_flag = 1, no_cabac_reset_flag = 1	dependent_slice_flag = 1,
방식 2	dependent_slice_flag = 0	dependent_slice_flag = 0
방식 3	dependent_slice_flag = 1, no_cabac_reset_flag = 0	dependent_slice_flag = 1, cabac_independent_flag = 1, tiles_or_entropy_coding_sync_idc = 3

[0283] 일 실시 예에서, 구문 요소 부(18)는 dependent_slice_flag에 의해 개별적으로 형성되나, 다른 실시 예에서, dependent_slice_flag 및 no_cabac_reset_flag의 조합이 구문 요소 부를 형성한다. 이전에 디코딩된 슬라이스의 저장된 상태들에 따라 심볼 확률들의 초기화가 관련되는 한 콘텍스트 변수들을 위한 동기화 과정이 참조된다. 특히, 디코더는 만일 last_ctb_cabac_init_flag=0이고 tiles_or_entropy_coding_sync_idc=2이면, 래

스터 스캔 순서에 따라 열 내의 두 번째 코딩 블록까지 이전에 디코딩된 슬라이스의 콘텍스트 적응적 엔트로피 디코딩에서 획득되는 것과 같은 심볼 확률들을 저장하고, 첫 번째 방식에 따라 현재 슬라이스의 콘텍스트 적응적 엔트로피 디코딩을 위한 심볼 확률들의 초기화에서, 현재 슬라이스와 관련된 코딩 블록들의 연속적인 서브셋들의 첫 번째 코딩 블록이 래스터 스캔 순서에 따른 열의 첫 번째 코딩 블록인지를 검사하도록 구성될 수 있으며, 만일 그렇다면, 래스터 스캔 순서에 따라 열의 두 번째 코딩 블록까지 이전에 디코딩된 슬라이스의 콘텍스트 적응적 엔트로피 디코딩에서 획득되는 것과 같은 저장된 심볼 확률들에 따라 현재 슬라이스의 콘텍스트 적응적 엔트로피 디코딩을 위한 심볼 확률들을 초기화하고, 만일 그렇지 않으면, 이전에 디코딩된 슬라이스의 끝까지 이전에 디코딩된 슬라이스의 콘텍스트 적응적 엔트로피 디코딩에서 획득되는 것과 같은 심볼 확률들에 따라 현재 슬라이스의 콘텍스트 적응적 엔트로피 디코딩을 위한 심볼 확률들을 초기화하도록 구성될 수 있다.

[0284] 따라서, 바꾸어 말하면, 구문을 위한 두 번째 실시 예에 따라, 디코더는 화상(10)이 세분화되는 슬라이스들(14)의 유닛들 내에서 화상이 코딩되는 데이터스트림(12)으로부터 화상(10)을 재구성할 수 있으며, 디코더는 적어도 두 가지 방식(20, 22)에 따라 현재 슬라이스를 디코딩하기 위하여 슬라이스 순서(16)에 따라 데이터스트림(12)으로부터 슬라이스들(14)을 디코딩하도록 구성되고 디코더는 적어도 두 가지 방식(20, 22) 중 어느 하나에 따라 현재 슬라이스를 디코딩하기 위하여 구문 요소 부(18), 즉 슬라이스들의 현재 슬라이스 내의 dependent_slice_flag에 응답한다. 적어도 두 가지 방식 중 첫 번째 방식(20)에 따라, 즉 만일 dependent_slice_flag=1이면, 디코더는 슬라이스 경계들을 가로지른 콘텍스트들의 도출, 콘텍스트들의 심볼 확률들의 연속적인 업데이트들 및 이전에 디코딩된 슬라이스의 심볼 확률들의 저장된 상태에 따른 심볼 확률들의 초기화(38, 40)를 포함하는 콘텍스트 적응적 엔트로피 디코딩(24), 및 슬라이스 경계들을 가로지른 예측 디코딩을 사용하여 데이터스트림(12)으로부터 현재 슬라이스를 디코딩하며, 적어도 두 가지 방식 중 두 번째 방식(22)에 따라, 즉 만일 dependent_slice_flag=0이면, 디코더는 슬라이스 경계들을 가로지르지 않도록 하기 위한 콘텍스트들의 도출의 제한, 콘텍스트들의 심볼 확률들의 연속적인 엡데이트 및 어떠한 이전에 디코딩된 슬라이스와 관계없는 심볼 확률들의 초기화를 갖는 콘텍스트 적응적 엔트로피 디코딩, 및 슬라이스 경계들을 가로지르지 않도록 하기 위한 예측 디코딩의 제한을 갖는 예측 디코딩을 사용하여 데이터스트림(12)으로부터 현재 슬라이스를 디코딩한다. 화상(10)은 열과 행들로 배치되고 서로 중에서 정의되는 래스터 스캔 순서(36)를 갖는 코딩 블록들(32)로 분할될 수 있으며, 디코더는 각각의 슬라이스(14)를 슬라이스 순서에 따른 래스터 스캔 순서(36)를 따라 서브셋들이 서로 뒤따르도록 하기 위하여 래스터 스캔 순서(36)로 코딩 블록들의 연속적인 서브셋들과 연관시키도록 구성된다. 디코더는 즉 tiles_or_entropy_coding_sync_idc=2에 응답하여, 래스터 스캔 순서(36)에 따라 열 내의 두 번째 코딩 블록(32)까지 이전에 디코딩된 슬라이스의 콘텍스트 적응적 엔트로피 디코딩에서 획득하는 것과 같이 심볼 확률들을 저장하고, 첫 번째 방식에 따라 현재 슬라이스의 콘텍스트 적응적 엔트로피 디코딩을 위한 심볼 확률들의 초기화에서, 현재 슬라이스와 관련된 코딩 블록들(32)의 연속적인 서브셋들의 첫 번째 코딩 블록이 래스터 스캔 순서에 따른 열의 첫 번째 코딩 블록(32)인지를 검사하도록 구성될 수 있으며, 만일 그렇다면, 래스터 스캔 순서(36)에 따라 열의 두 번째 코딩 블록까지 이전에 디코딩된 슬라이스의 콘텍스트 적응적 엔트로피 디코딩에서 획득되는 것과 같은 저장된 심볼 확률들에 따라 현재 슬라이스의 콘텍스트 적응적 엔트로피 디코딩을 위한 심볼 확률들을 초기화하고(40) 만일 그렇지 않으면, 이전에 디코딩된 슬라이스의 끝까지 이전에 디코딩된 슬라이스를 콘텍스트 적응적 엔트로피 디코딩에서 획득되는 것과 같은 심볼 확률들에 따라 현재 슬라이스의 콘텍스트 적응적 엔트로피 디코딩을 위한 심볼 확률들을 초기화하도록(38) 구성될 수 있다. 디코더는 적어도 3가지 방식 중 어느 하나, 즉 첫 번째(20) 및 세 번째(42) 또는 두 번째 방식(22) 중 어느 하나에 따라 현재 슬라이스를 디코딩하기 위하여, 슬라이스들의 현재 슬라이스 내의 구문 요소 부(18)에 응답하도록 구성될 수 있으며, 디코더는 즉 만일 dependent_slice_flag=1이고 tiles_or_entropy_coding_sync_idc=3이면, 세 번째 방식(42)에 따라 슬라이스 경계들을 가로지르지 않도록 하기 위한 콘텍스트들의 도출의 제한, 어떠한 이전에 디코딩된 슬라이스와 관계없는 콘텍스트들의 심볼 확률들의 연속적인 업데이트 및 심볼 확률들의 초기화를 갖는 콘텍스트 적응적 엔트로피 디코딩, 및 슬라이스 경계들을 가로지르는 예측 디코딩을 사용하여 데이터스트림으로부터 현재 슬라이스를 디코딩하도록 구성되며. 첫 번째 및 세 번째 방식 중 하나는 구문 요소, 즉 cabac_independent_flag에 따라 선택된다. 디코더는 즉 만일 tiles_or_entropy_coding_sync_idc=0, 1, 및 3이면(cabac_independent flag=0일 때 "3"), 이전에 디코딩된 슬라이스의 끝까지 이전에 디코딩된 슬라이스를 콘텍스트 적응적 엔트로피 디코딩에서 획득되는 것과 같은 심볼 확률들을 저장하고, 저장된 심볼 확률들에 따라 현재 슬라이스의 콘텍스트 적응적 엔트로피 디코딩을 위한 심볼 확률들을 초기화하도록 구성될 수 있다. 디코더는 즉, 만일 tiles_or_entropy_coding_sync_idc=1이면, 첫 번째 및 두 번째 방식에서, 화상이 세분화되는 타일들 내의 예측 디코딩을 제한하도록 구성될 수 있다.

[0285] 자연적으로, 인코더는 디코더가 위에 설명된 장점들을 획득하는 것을 가능하게 하기 위하여 위의 설명된 구문을

설정할 수 있다. 인코더는 다중 코어, 인코더와 같은, 병렬 인코더일 수 있으나, 반드시 그럴 필요는 없다. 슬라이스들(14)의 유닛들 내의 데이터스트림 내로의 화상(10)의 인코딩을 위하여, 인코더는 슬라이스 순서(16)에 따라 데이터스트림(12) 내로 슬라이스들(14)을 인코딩할 수 있다. 인코더는 슬라이스들의 현재 슬라이스를 위한 구문 요소 부(18)를 결정할 수 있고, 구문 요소 부가 두 가지 방식(20, 22) 중 적어도 하나에 따라 코딩되는 현재 슬라이스를 시그널링하도록 이를 슬라이스들의 현재 슬라이스 내로 코딩할 수 있으며, 만일 현재 슬라이스가 적어도 두 가지 방식 중 첫 번째 방식(20)에 따라 코딩되면, 슬라이스 경계들을 가로질러 콘텍스트들의 도출, 콘텍스트들의 심볼 확률들의 연속적인 업데이트 및 이전에 인코딩된 슬라이스의 심볼 확률들의 저장된 상태에 따라 심볼 확률들의 초기화(38, 40)를 포함하는 콘텍스트 적응적 엔트로피 인코딩(24)을 사용하여 현재 슬라이스를 데이터스트림(12) 내로 인코딩할 수 있으며, 만일 현재 슬라이스가 적어도 두 가지 방식 중 두 번째 방식(40)에 따라 코딩되면, 슬라이스 경계들을 가로지르기 위한 콘텍스트들의 도출의 제한, 콘텍스트들의 심볼 확률들의 연속적인 업데이트 및 이전에 인코딩된 슬라이스의 심볼 확률들의 저장된 상태에 따른 심볼 확률들의 초기화를 갖는 콘텍스트 적응적 엔트로피 인코딩(24), 및 슬라이스 경계들을 가로지르지 않도록 예측 인코딩의 제한을 갖는 예측 코딩을 사용하여 현재 슬라이스를 데이터스트림(12) 내로 인코딩할 수 있다. 화상(10) 열들과 행들 내에 배치되고 서로 간에서 정의되는 래스터 스캔 순서(36)를 갖는 코딩 블록들(32)로 분할될 수 있으나, 인코더는 각각의 슬라이스를, 서브셋들이 슬라이스 순서에 따른 래스터 스캔 순서(36)를 따라 서로 따르도록 하기 위하여 래스터 스캔 순서(36) 내의 코딩 블록들(32)의 연속적인 서브셋과 관련시키도록 구성될 수 있다. 인코더는 래스터 스캔 순서(36)에 따라 열 내의 두 번째 코딩 블록(32)까지 이전에 인토딩된 슬라이스의 콘텍스트 적응적 엔트로피 인코딩에서 획득되는 것과 같은 심볼 확률들을 저장하도록 구성될 수 있고, 첫 번째 방식에 따라 현재 슬라이스의 콘텍스트 적응적 엔트로피 인코딩을 위한 심볼 확률들의 초기화에서, 현재 슬라이스와 관련된 코딩 블록들(32)의 연속적인 서브셋의 첫 번째 코딩 블록이 래스터 스캔 순서에 따른 열 내의 첫 번째 코딩 블록인지에 대하여 검사하도록 구성될 수 있으며, 만일 그렇다면, 래스터 스캔 순서(36)에 따라 열 내의 두 번째 코딩 블록까지 이전에 인코딩된 슬라이스의 콘텍스트 적응적 엔트로피 인코딩에서 획득된 것과 같이 저장된 심볼 확률들에 따라 현재 슬라이스의 콘텍스트 적응적 엔트로피 인코딩을 위한 심볼 확률들을 초기화하도록(40) 구성될 수 있고, 만일 그렇지 않으면, 이전에 인코딩된 슬라이스의 끝까지 이전에 디코딩된 슬라이스의 콘텍스트 적응적 엔트로피 인코딩에서 획득된 것과 같이 심볼 확률들에 따라 현재 슬라이스의 콘텍스트 적응적 엔트로피 인코딩을 위하여 심볼 확률들을 초기화하도록(38) 구성될 수 있다. 인코더는 현재 슬라이스가 적어도 3가지 방식 중 하나에 따라, 즉, 첫 번째(20) 및 세 번째 방식(42) 또는 두 번째 방식(20)으로 그것들 내로 코딩되도록 시그널링되기 위하여 슬라이스들(14)의 현재 슬라이스 내로 구문 요소 부(18)를 코딩하도록 구성될 수 있으며, 인코더는 세 번째 방식(42)에 따라, 슬라이스 경계들을 가로지르지 않도록 하기 위한 콘텍스트의 도출의 제한, 콘텍스트들의 심볼 확률들의 연속적인 업데이트 및 어떠한 이전에 인코딩된 슬라이스와 관계없는 심볼 확률들의 초기화를 갖는 콘텍스트 적응적 엔트로피 인코딩, 및 슬라이스 경계들을 가로지르는 예측 코딩을 사용하여 현재 슬라이스를 데이터스트림 내로 인코딩하도록 구성되며, 인코더는 구문 요소, 즉 예를 들면 cabac_independent_flag를 사용하여 첫 번째 및 세 번째 방식 중 어느 하나 사이를 구별한다. 인코더는 dependent_slices_present_flag와 같은 제네릭(generic) 구문 요소를 결정하도록 구성될 수 있고, 제네릭 구문 요소에 따른 적어도 두 가지 제네릭 작동 방식 중 하나에서의 작동으로, 즉, 각각의 슬라이스를 위하여 구문 요소 부의 코딩을 실행하는, 첫 번째 제네릭 작동 방식, 및 첫 번째 방식 이외의 적어도 두 가지 방식 중 다른 하나를 필연적으로 사용하는, 두 번째 제네릭 작동 방식에 따라, 데이터스트림 내로 이를 기록하도록(write) 구성될 수 있다. 인코더는 첫 번째 및 두 번째 방식에 따라 필연적으로 또는 중단됨이 없이, 현재 슬라이스의 시작부터 끝까지 심볼 확률들을 연속적으로 계속해서 업데이트하도록 구성될 수 있다. 인코더는 이전에 인코딩된 슬라이스의 끝까지 이전에 인코딩된 슬라이스를 콘텍스트 적응적 엔트로피 인코딩 내에서 획득되는 것과 같은 심볼 확률들을 저장하고, 첫 번째 방식에 따른 현재 슬라이스의 콘텍스트 적응적 엔트로피 인코딩을 위한 심볼 확률들을 초기화하는데 있어서, 저장된 심볼 확률들에 따라 현재 슬라이스의 콘텍스트 적응적 엔트로피 인코딩을 위한 심볼 확률들을 초기화하도록 구성될 수 있다. 그리고 인코더는 첫 번째 및 두 번째 방식으로, 화상이 세분화되는 타일들 내로의 예측 인코딩을 제한할 수 있다.

[0286] 인코더의 가능한 구조가 완전성을 위하여 도 26에 도시된다. 예측기(70)는 예측기(28)와 거의 동일하게 작동하는데, 즉 예측을 실행하나, 또한 최적화에 의해, 예를 들면 예측 파라미터들과 방식들을 포함하는 코딩 파라미터들을 결정한다. 모듈들(26 및 27)이 또한 디코더 내에 발생한다. 감산기(72)는 변환 및 양자화 모듈(74) 내에 손실 코딩되는 (양자화의 사용에 의해, 그리고 선택적으로 스펙트럼 분해 변환을 사용하여) 무손실 예측 잔류를 결정한다. 엔트로피 코더(76)는 콘텍스트 적응적 엔트로피 인코딩을 실행한다.

[0287] 위의 구체적인 구문 예들에 더하여, 서로 다른 실시 예가 이후에 사용되는 용어들과 위에서 사용된 용어들 사이

의 유사성의 표현과 함께 아래에 설명된다.

[0288] 특히, 위에서의 특별한 설명 없이도, 종속 슬라이스들은 외부로부터 알려진 지식의 이용을 허용한다는 점에서 "종속적"이지 않으며 그것의 경계는 위에 설명된 것과 같이, 빠르게 적응되는 엔트로피 콘텍스트들을 갖거나 또는 그것의 경계의 크로싱의 허용에 기인하여 더 나은 공간 예측을 달성한다. 오히려, 화상을 슬라이스들로 분할함으로써 슬라이스 헤더들을 정의하기 위하여 소비되어야만 하는 레이트 비용을 절약하기 위하여, 종속 슬라이스들은 이전의 슬라이스들로부터의 슬라이스 헤더 구분의 일부분을 채택하는데, 즉 이러한 슬라이스 헤더 구문 부는 종속 슬라이스들을 위하여 다시 전송되지 않는다. 이는 예를 들면 이전의 슬라이스로부터 어떠한 슬라이스 형태가 채택되는지에 따라, 예를 들면, 도 16의 100 및 도 21의 102에 도시된다. 이러한 측정에 의해, 종속 슬라이스 및 독립 슬라이스와 같은, 슬라이스들로의 화상의 세분화는 비싼 비트 소모와 관련하여 덜 비싸다.

[0289] 이는 아래에 설명되는 실시 예에서, 약간 다른 용어 선택에 이르게 하는 방금 언급된 종속성이며, 슬라이스들은 슬라이스 헤더 구문이 개별적으로 설정될 수 있는 화상의 유닛 부들로서 정의된다. 따라서, 슬라이스들은 이제 독립 슬라이스 세그먼트로 불리는, 하나의 독립/규칙/정상 슬라이스(위의 명칭을 사용하여) 및 이제 종속 슬라이스 세그먼트로 불리는, 0, 하나 또는 그 이상의 종속 슬라이스(위의 명칭을 사용하여)로 구성된다.

[0290] 예를 들면, 도 27은 두 개의 슬라이스로 분할되는 화상을 도시하는데, 하나는 슬라이스 세그먼트들(14_1 내지 14_3)에 의해 형성되고 나머지는 단독으로 슬라이스 세그먼트(14_4)에 의해 형성된다. 지수들(1 내지 4)은 코딩 순서 내의 슬라이스 순서를 나타낸다. 도 28a 및 b는 화상(10)의 두 개의 타일로의 세분화의 경우에서의 서로 다른 예를 도시하는데, 도 28a의 경우에, 타일들(50_1 및 50_2)을 모두 포함하는, 모든 5개의 슬라이스 세그먼트(14)에 의해 형성되고(지수는 코딩 순서 내에서 다시 상승), 도 28b의 경우에, 두 개의 슬라이스는 각각 타일(50_1)을 포함하는, 슬라이스 세그먼트들(14_1, 14_2, 14_3 및 14_4)에 형성되며, 나머지 슬라이스는 타일(50_2)을 포함하는, 슬라이스 세그먼트들(14_5-14_6)에 의해 형성된다.

[0291] 정의들은 다음과 같을 수 있다:

[0292] **종속 슬라이스 세그먼트**: 슬라이스 세그먼트 헤더의 일부 구문 요소들의 값들이 이전에, 위의 실시 예들에서 종속 슬라이스로 불리는, 디코딩 순서 내의 선행의 독립 슬라이스 세그먼트를 위한 값들로부터 추론되는 슬라이스 세그먼트.

[0293] **독립 슬라이스 세그먼트**: 슬라이스 세그먼트 헤더의 일부 구문 요소들의 값들이 이전에, 위의 실시 예들에서 정상 슬라이스로 불리는, 디코딩 순서 내의 선행의 슬라이스 세그먼트를 위한 값들로부터 추론되지 않는 슬라이스 세그먼트.

[0294] **슬라이스**: 하나의 독립 슬라이스 세그먼트 및 동일한 액세스 유닛/화상 내의 그 다음의 독립 슬라이스 세그먼트를 선행하는 모든 뒤따르는 종속 세그먼트(만일 있으면) 내에 포함되는 코딩 트리 유닛들의 정수.

[0295] **슬라이스 헤더**: 현재 슬라이스 세그먼트이거나 또는 현재 종속 슬라이스 세그먼트를 선행하는 독립 슬라이스 세그먼트인 독립 슬라이스 세그먼트의 슬라이스 세그먼트 헤더.

[0296] **슬라이스 세그먼트**: 타일 스캔 내에 연속적으로 정렬되고 단일 네트워크 추상 계층 유닛 내에 포함되는 코딩 트리 유닛들의 정수; 각각의 화상의 슬라이스 세그먼트들로의 세분이 분할이다.

[0297] **슬라이스 세그먼트 헤더**: 슬라이스 세그먼트 내에 표현되는 첫 번째 또는 모든 코딩 트리 유닛과 관련된 데이터 요소들을 포함하는 코딩된 슬라이스 세그먼트의 일부분.

[0298] "방식들(20 및 22)", 즉 "종속 슬라이스 세그먼트" 및 "독립 슬라이스 세그먼트"의 시그널링은 다음과 같을 수 있다:

[0299] 화상 파라미터 세트와 같은 일부 추가의 네트워크 추상 계층 유닛들에서, 구문 요소는 특정 화상들을 위한 시퀀스의 특정 화상을 위하여 종속 슬라이스가 사용되는지 그렇지 않은지에 대하여 시그널링하도록 사용될 수 있다:

[0300] 1과 동일한 **dependent_slice_segments_enables_flag**는 슬라이스 세그먼트 헤더들 내의 구문 요소(dependent_slice_segement_flag)의 존재를 지정한다. 0과 동일한 dependent_slice_segments_enables_flag는 슬라이스 세그먼트 헤더들 내의 구문 요소(dependent_slice_segement_flag)의 부재를 지정한다.

[0301] **dependent_slice_segments_enables_flag**는 이전에 설명된 **dependent_slices_present_flag**의 범위와 유사하다.

[0302] 유사하게, **dependent_slice_flag**는 슬라이스들과 관련하여 서로 명명법을 설명하기 위하여 **dependent_slice_segment_flag**로 불린다.

[0303] 0과 동일한 **dependent_slice_segment_flag**는 현재 슬라이스 세그먼트의 헤더 내에 존재하지 않는 각각의 슬라이스 세그먼트 헤더 구문 요소의 값이 슬라이스 헤더, 즉 선행 독립 슬라이스 세그먼트의 슬라이스 세그먼트 헤더 내의 상응하는 슬라이스 세그먼트 헤더 구문 요소의 값과 동일한 것으로 추론되는 것을 지정한다.

[0304] 화상 레벨과 같은, 동일한 레벨에서, 다음의 구문 요소가 포함될 수 있다:

[0305] 1과 동일한 **entropy_coding_sync_enabled_flag**는 콘텍스트 변수들을 위한 특정 동기화 과정이 화상 파라미터 세트(PPS)를 언급하는 각각의 화상 내의 각각의 타일 내의 코딩 트리 블록들의 하나의 열의 첫 번째 코딩 트리 블록을 포함하는 코딩 트리 유닛을 디코딩하기 전에 적용되고, 콘텍스트 변수들을 위한 특정 저장 과정이 화상 파라미터 세트를 언급하는 각각의 화상 내의 각각의 타일 내의 코딩 트리 블록들의 하나의 열의 두 번째 코딩 트리 블록을 포함하는 코딩 트리 유닛을 디코딩하기 전에 적용되는 것을 지정한다. 0과 동일한 entropy_coding_sync_enabled_flag는 콘텍스트 변수들을 위한 어떠한 특정 동기화 과정도 화상 파라미터 세트를 언급하는 각각의 화상 내의 각각의 타일 내의 코딩 트리 블록들의 하나의 열의 첫 번째 코딩 트리 블록을 포함하는 코딩 트리 유닛을 디코딩하기 전에 적용되지 않고, 콘텍스트 변수들을 위한 어떠한 특정 저장 과정도 화상 파라미터 세트를 언급하는 각각의 화상 내의 각각의 타일 내의 코딩 트리 블록들의 하나의 열의 두 번째 코딩 트리 블록을 포함하는 코딩 트리 유닛을 디코딩하기 전에 적용되지 않는 것을 지정한다.

[0306] entropy_coding_sync_enabled_flag의 값이 코딩된 비디오 시퀀스 내에서 작동되는 모든 화상 파라미터 세트를 위하여 동일해야만 하는 것이 비트스트림 적합성의 요구조건이다.

[0307] entropy-config_sync_enabled_flag가 1과 동일하고 슬라이스 내의 첫 번째 코딩 트리 블록이 타일 내의 코딩 트리 블록들의 하나의 열의 첫 번째 코딩 트리 블록이 아닐 때, 슬라이스 내의 마지막 코딩 트리 블록이 슬라이스 내의 첫 번째 코딩 트리 블록과 동일한 코딩 트리 블록들의 열에 속해야만 하는 것이 비트스트림 적합성의 요구조건이다. entropy-config_sync_enabled_flag가 1과 동일하고 슬라이스 세그먼트 내의 첫 번째 코딩 트리 블록이 타일 내의 코딩 트리 블록들의 하나의 열의 첫 번째 코딩 트리 블록이 아닐 때, 슬라이스 세그먼트 내의 마지막 코딩 트리 블록이 슬라이스 세그먼트 내의 첫 번째 코딩 트리 블록과 동일한 코딩 트리 블록들의 열에 속해야만 하는 것이 비트스트림 적합성의 요구조건이다.

[0308] 이미 설명된 것과 같이, 코딩 트리 블록들(30) 중에서의 코딩/디코딩 순서는 첫 번째 타일의 스캐닝으로 시작하여 래스터 방식으로 상부로부터 하부로의 열 방식에 이르게 하고 그리고 나서 만일 하나 이상의 타일이 화상 내에 존재하면, 그 다음의 타일을 방문한다.

[0309] 디코더(5, 및 따라서 인코더)는 화상의 엔트로피 디코딩(코딩) 슬라이스 세그먼트들(14) 내에서 다음과 같이 작용한다:

[0310] A1) 현재 디코딩되는/코딩되는 구문 요소(synE1)가 타일(50), 슬라이스 세그먼트(14) 또는 코딩 트리 블록의 열의 첫 번째 구문 요소일 때마다, 도 29의 초기화 과정이 시작된다.

[0311] A2) 그렇지 않으면, 이러한 구문 요소의 디코딩은 현재 엔트로피 콘텍스트들을 사용하여 일어난다.

[0312] A3) 만일 현재 구문 요소가 코딩 트리 블록(CTB, 30) 내의 마지막 구문 요소이었으면, 도 30에 도시된 것과 같이 엔트로피 콘텍스트 저장 과정이 시작된다.

[0313] A4) 과정은 그 다음의 구문 요소로 A1)에서 진행된다.

[0314] 초기화 과정에서, synE1이 슬라이스 세그먼트(14) 또는 타일(50)의 첫 번째 구문 요소인지가 검사된다(200). 만일 그렇다면, 단계 202에서 콘텍스트들은 어떠한 이전의 슬라이스 세그먼트와 관계없이 초기화된다. 만일 그렇지 않으면, synE1이 코딩 트리 블록들(30)의 열의 첫 번째 구문 요소인지, 그리고 **entropy-coding_sync_enabled_flag**가 1과 동일한지가 검사된다(204). 만일 그렇다면, 동일한 타일의 코딩 트리 블록들(30)의 이전의 라인에서, 두 번째 코딩 트리 블록(30)이 이용가능한지가 검사된다(206, 도 23 참조). 만일 그렇다면, 단계 210에서 형태 40 채택을 위한 현재 저장된 콘텍스트 확률들을 사용하여 40에 따른 콘텍스트 채택이 실행된다. 만일 그렇지 않으면, 콘텍스트들을 단계 202에서의 어떠한 이전의 슬라이스 세그먼트와 관계없이 초기화된다. 만일 검사(204)가 아니 것으로 나타나면, 단계 212에서 synE1이 종속 슬라이스 세그먼트(14)의 첫 번째 코딩 트리 블록 내의 첫 번째 구문 요소인지, 그리고 dependent_slice_flag가 1과 동일한지가 검사되고, 만일 그렇다면, 단계 214에서 형태 38 채택을 위한 현재 저장된 콘텍스트 확률들을 사용하여 38에 따른 콘텍스트

채택이 실행된다.214, 212, 210 및 202 단계 중 어느 하나의 단계 후에, 디코딩/코딩이 실제로 시작된다.

[0315] 1과 동일한 dependent_slice_segment_flag를 갖는 종속 슬라이스 세그먼트들은 따라서 코딩 효율 불이익이 거의 없이 코딩/디코딩 지연을 더 감소시키는데 도움을 준다.

[0316] 도 30의 저장 과정에서, 단계 300에서 코딩된/디코딩된 synE1이 코딩 트리 블록들(30)의 열의 두 번째 코딩 트리 블록(30)의 마지막 구문 요소인지, 그리고 entropy_coding_sync_enabled_flag가 1과 동일한지가 검사된다. 만일 그렇다면, 단계 302에서 현재 엔트로피 콘텍스트들, 즉, 40과 같은 채택에 특이적인 저장장치 내의, 콘텍스트들의 엔트로피 코딩 확률들이 저장된다. 유사하게, 단계 304에서, 단계 300 또는 302에 더하여, 코딩된/디코딩된 synE1이 슬라이스 세그먼트(14)의 마지막 구문 요소인지, 그리고 dependent_slice_segment_flag가 1과 동일한지가 검사된다. 만일 그렇다면, 단계 306에서 현재 엔트로피 콘텍스트들, 즉, 40과 같은 채택에 특이적인 저장장치 내의, 콘텍스트들의 엔트로피 코딩 확률들이 저장된다.

[0317] 구문 요소가 코딩 트리 블록 열의 첫 번째 synE1인지에 대하여 문의하는(queryng) 어떠한 검사는 예를 들면, 슬라이스 세그먼트의 헤더들 내의 구문 요소(slice_adress, 400), 즉, 디코딩 순서를 따라 각각의 슬라이스 세그먼트의 시작의 위치를 나타내는 시작 구문 요소를 이용한다는 것에 유의하여야 한다.

[0318] 파면 병렬 처리 과정을 사용하여 데이터스트림(12)으로부터 화상(10)을 재구성하는 경우에 있어서, 디코더는 파면 병렬 처리 서브스트림 엔트리 지점들을 검색하기 위하여 후자의 출발 구문 부(400)를 정확히 사용할 수 있다. 각각의 슬라이스 세그먼트가 화상(10) 내의 각각의 슬라이스 세그먼트의 디코딩 시작의 위치를 나타내는 출발 구문 부(400)를 포함하기 때문에, 디코더는 슬라이스 세그먼트의 출발 구문 부들(400)을 사용하여 슬라이스 세그먼트들이 그룹핑되는(grouped) 파면 병렬 처리 서브스트림들의 엔트리 지점들을 식별할 수 있으며, 슬라이스 세그먼트들은 화상의 왼쪽 면에서 시작한다. 그리고 나서, 디코더는 슬라이스 순서에 따라 파면 병렬 처리 서브스트림들의 디코딩을 순차적으로 시작하는 스태그드 방식으로 파면 병렬 처리 서브스트림들을 병렬로 디코딩할 수 있다. 슬라이스 세그먼트들은 심지어 하나의 화상 폭, 즉 코딩 트리 블록들의 하나의 열보다 작을 수 있으며, 따라서 그것들의 전송은 전체 전송 종단간 지연을 더 감소시키기 위하여 파면 병렬 처리 서브스트림 중간에 삽입될(interleaved) 수 있다. 인코더는 각각의 슬라이스(14)에 화상(10) 내의 각각의 슬라이스의 코딩 시작의 위치를 나타내는 출발 구문 부(400)를 제공하고 슬라이스들을 파면 병렬 처리 서브스트림들로 그룹핑할 수 있으며, 따라서 각각의 파면 병렬 처리 서브스트림을 위하여, 슬라이스 순서 내의 첫 번째 슬라이스는 화성의 왼쪽 면에서 시작한다. 인코더는 심지어 자체로, 화상을 인코딩하는데 파면 병렬 처리 과정을 사용할 수 있으며; 인코더는 슬라이스 순서에 따라 파면 병렬 처리 서브스트림들의 인코딩을 순차적으로 시작하는 스태그드 방식으로 파면 병렬 처리 서브스트림들을 병렬 인코딩한다.

[0319] 그런데, 파면 병렬 처리 서브스트림들의 엔트리 지점들을 위치시키기 위한 수단으로서 슬라이스 세그먼트의 출발 구문 부들을 사용하는 후자의 양상은 종속 슬라이스 개념 없이 사용될 수 있다.

[0320] 다음과 같이 위의 변수들을 설정함으로써, 병렬 처리 화상(10)을 위하여 모두 실현 가능할 수 있다:

	tiles_enabled_flag==0	tiles_enabled_flag==1	entropy_cooding_sync_enable_flag ==1	tiles_enabled_flag==1	entropy_cooding_sync_enable_flag ==1
dependent_slice_segment_flag	0/1	0/1	0/1	0/1	0/1
타일들로의 화상 세분화	아니오	예	아니오	예	아니오
파면 병렬 처리 서브스트림 또는 타일의 시작을 나타내기 위한 엔트리 지점들을 갖는 슬라이스 세분화	아니오	예	예	아니오	아니오
가능한 병렬 처리	슬라이스들 내	타일들 내	파면 병렬 처리	타일들 내	파면 병렬 처리

[0321]

[0322] 심지어 파면 병렬 처리를 타일 분할과 믹싱하는 것이 실현 가능할 수 있다. 그러한 경우에 있어서, 파면 병렬 처리는 타일들을 개별 화상들로서 취급할 수 있으며, 파면 병렬처리를 사용하는 각각은 하나 또는 그 이상의 종

속 슬라이스 세그먼트를 갖는 슬라이스로 구성될 수 있으며, 단계 300 및 208에서의 검사는 단계 204에서와 같이, 동일한 타일 내의 위의 코딩 트리 블록 열의 두 번째 코딩 트리 블록를 언급할 수 있으며, A1은 현재 타일의 코딩 트리 블록(30) 열 내의 첫 번째 코딩 트리 블록(30)을 언급할 수 있다. 그러한 경우에 있어서, 위의 테이블은 확장될 수 있다:

	tiles_enabled_flag	tiles_enabled_flag==1	entropy_coodinng_sync_enable_flag ==1	tiles_enabled_flag==1	entropy_coodinng_sync_enable_flag ==1	entropy_coodinng_sync_enable_flag ==1	entropy_coodinng_sync_enable_flag ==1
dependent_slice_segment_flag	0/1	0/1	0/1	0/1	0/1	0/1	0/1
타일들로의 화상 세분화	아니오	예	아니오	예	아니오	예	예
파면 병렬 처리 서브스트림 또는 타일의 시작을 나타내기 위한 엔트리 지점들을 갖는 슬라이스 세분화	아니오	예	예	아니오	아니오	아니오 (고유의 종속 슬라이스 내의 타일의 각각의 서브스트림)	예 (타일의 슬라이스 내의 파면 병렬 처리 서브스트림들을 위한 개별 슬라이스 내의 각각의 타일및 엔트리 지점들
가능한 병렬 처리	슬라이스들 내	타일들 내	파면 병렬 처리	타일들 내	파면 병렬 처리	타일들 내의 파면 병렬 처리 서브스트림들	타일들 내의 파면 병렬 처리 서브스트림들

[0323]

[0324] 간단히 설명하면, 후자의 확장은 실시 예 2로 가능하였을 수 있다. 실시 예 2는 다음의 처리를 허용한다:

	구문 레벨				
tiles_or_entropy_coding_sync_idc	화상 당	0	1	2	3
cabac_independent_flag	슬라이스 당	존재하지 않음	존재하지 않음	존재하지 않음	0/1
dependent_slice_flag	슬라이스 당	0/1	0/1	0/1	0/1
타일들로의 화상 세분화		아니오	예	아니오	아니오
파면 병렬 처리 서브스트림들 또는 타일들의 시작을 나타내기 위한 엔트리 지점들로의 슬라이스 세분화		아니오	예	예	아니오
가능한 병렬 처리		슬라이스 내	타일들 내	파면 병렬 처리	슬라이스들의 엔트로피 코딩

[0325]

[0326] 그러나 다음의 확장들과 함께, 아래의 테이블은 다음을 야기할 수 있다:

[0327] 화상 파라미터 세트의 시멘틱에 추가:

[0328] 만일 tiles_or_entropy_coding_sync_idc가 4와 동일하면, 각각, 그러나 코딩 트리 블록들의 첫 번째 열은 1로 설정된 종속 슬라이스 세트와 다른 슬라이스 내에 포함되어야만 한다. 서로 다른 열들의 코딩 트리 블록들은 동일한 슬라이스 내에 존재해서는 안 된다. 코딩 트리 블록 열 당 하나 이상의 슬라이스가 존재할 수 있다.

[0329] 만일 tiles_or_entropy_coding_sync_idc가 5와 동일하면, 각각, 그러나 코딩 트리 블록들의 첫 번째 열은 서로 다른 슬라이스 내에 포함되어야만 한다. 서로 다른 타일들의 코딩 트리 블록들은 동일한 슬라이스 내에 존재해서는 안 된다. 타일 당 하나 이상의 슬라이스가 존재할 수 있다.

[0330] 또 다른 설명을 위하여 도 31이 참조된다.

[0331] 즉, 위의 테이블은 확장될 수 있다:

[0332]

	구문 x 레벨								
tiles_or_entropy_coding_sync_idc	화상당	0	1	2	3	5	4	6	7
cabac_independent_flag	슬라이스 당	존재하지 않음	존재하지 않음	존재하지 않음	0/1	존재하지 않음	존재하지 않음	존재하지 않음	존재하지 않음
dependent_slice_flag	슬라이스 당	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
타일들로의 화상 세분화		아니오	예	아니오	아니오	예	아니오	예	예
Slice subdivision with entry points to indicate start of WPP substream or Tile. 파면 병렬 처리 서브스트림들 또는 타일들의 시작을 나타내기 위한 엔트리 지점들로의 슬라이스 세분화		아니오	예	예	아니오	아니오	아니오	아니오(고유의 종속 슬라이스 내의 타일의 각각의 서브스트림)	예 (개별 슬라이스 내의 각각의 타일 및 타일의 슬라이스 내의 WPP 서브스트림을 위한 엔트리 지점
가능한 병렬 처리		슬라이스 내	타일들 내	파면 병렬 처리	슬라이스들의 엔트로피 코딩	타일들 내	파면 병렬 처리	타일들 내의 파면 병렬 처리 서브스트림들	타일들 내의 파면 병렬 처리 서브스트림들

[0333] 위의 실시 예들과 관련하여, 디코더는 예를 들면, 첫 번째 또는 두 번째 방식으로, tiles_or_entropy_coding_sync_idc=1.2에 응답하여, 현재 슬라이스의 병렬 서브섹션들로의 세분화를 나타내는 현재 슬라이스로부터의 정보를 판독하도록 구성될 수 있다는 것을 이해하여야 하며, 병렬 서브섹션들은 첫 번째 병렬 서브섹션의 끝에서 콘텍스트 적응적 엔트로피 디코딩을 중단하고, 첫 번째 방식에서, 선행 병렬 서브섹션의 심볼 확률들의 저장된 상태에 따른 심볼 확률들의 초기화 및, 두 번째 방식에서, 어떠한 이전에 디코딩된 슬라이스 및 어떠한 이전에 디코딩된 병렬 서브섹션과 관계없는 심볼 확률들의 초기화를 포함하는 어떠한 뒤따르는 병렬 서브섹션의 시작에서 콘텍스트 적응적 엔트로피 디코딩을 새로 재개하는, 파면 병렬 처리 서브스트림들 또는 타일들일 수 있다.

[0334] 따라서, 위의 설명은 타일들, 파면 병렬 처리 서브스트림들, 슬라이스들 또는 엔트로피 슬라이스들 내에 구조화되는 것과 같이, 새로운 HEVC 코딩 표준에 의해 제공되는 것과 같이 구조화된 비디오 데이터의 저지연 인코딩, 디코딩, 캡슐화 및 전송을 위한 방법들을 나타내었다.

[0335] 특히, 인코딩, 디코딩 및 전송 과정에서의 최소 레이턴시를 얻기 위하여 종래의 시나리오에서 병렬 인코딩된 데이터의 전송 방법이 정의되었다.따라서, 게이밍, 원격 수술 등과 같은 최소 지연 적용들을 허용하기 위하여 파이프라인식 병렬 코딩, 전송 및 디코딩 접근법이 설명되었다.

[0336] 게다가, 위의 실시 예들은 저지연 전송 시나리오에서 사용하도록 하기 위하여 파면 병렬 처리의 격차를 해소하였다. 따라서, 파면 병렬 처리 서브스트림들을 위한 새로운 캡슐화 포맷, 종속 슬라이스를 설명하였다. 이러한 종속 슬라이스는 엔트로피 슬라이스 데이터, 파면 병렬 처리 서브스트림, 최대 코딩 유닛들의 전체 열, 슬라이

스의 단지 하나의 단편을 포함할 수 있으며, 이전에 전송된 슬라이스 헤더는 또한 포함된 단편 데이터에 적용된다. 포함된 데이터는 서브 슬라이스 헤더 내에 시그널링된다.

[0337] 새로운 슬라이스들의 명칭은 또한 "서브셋/경량 슬라이스들"일 수 있으나, 명칭 "종속 슬라이스"가 더 나은 것으로 알려졌다.

[0338] 코딩 및 전송에서의 병렬화의 레벨을 설명하는 시그널링이 설명되었다.

[0339] 장치의 맥락에서 일부 양상들이 설명되었으나, 이러한 양상들은 또한 블록 또는 장치가 방법 단계 또는 방법 단계의 특징과 상응하는, 상응하는 방법의 설명을 나타낸다는 것은 자명하다. 유사하게, 방법 단계의 맥락에서 설명된 양상들은 또한 상응하는 블록 아이템 혹은 상응하는 장치의 특징을 나타낸다. 일부 또는 모든 방법 단계는 예를 들면, 마이크로프로세서, 프로그램가능 컴퓨터 또는 전자 회로 같은 하드웨어 장치에 의해(또는 사용하여) 실행될 수 있다. 일부 실시 예들에서, 일부 하나 또는 그 이상의 가장 중요한 방법 단계는 그러한 장치에 의해 실행될 수 있다.

[0340] 특정 구현 요구사항들에 따라, 본 발명의 실시 예는 하드웨어 또는 소프트웨어에서 구현될 수 있다. 구현은 디지털 저장 매체, 예를 들면, 그 안에 저장되는 전자적으로 판독가능한 제어 신호들을 갖는, 플로피 디스크, DVD, 블루-레이, CD, RON, PROM, EPROM, EEPROM 또는 플래시 메모리를 사용하여 실행될 수 있으며, 이는 각각의 방법이 실행되는 것과 같이 프로그램가능 컴퓨터 시스템과 협력한다(또는 협력할 수 있다). 따라서, 디지털 저장 매체는 컴퓨터로 판독가능할 수 있다.

[0341] 본 발명에 따른 일부 실시 예들은 여기에 설명된 방법들 중 어느 하나가 실행되는 것과 같이, 프로그램가능 컴퓨터 시스템과 협력할 수 있는, 전자적으로 판독가능한 제어 신호들을 갖는 데이터 캐리어를 포함한다.

[0342] 일반적으로, 본 발명의 실시 예들은 프로그램 코드를 갖는 컴퓨터 프로그램 제품으로서 구현될 수 있으며, 프로그램 코드는 컴퓨터 프로그램 제품이 컴퓨터 상에서 구동할 때 방법들 중 어느 하나를 실행하도록 운영될 수 있다. 프로그램 코드는 예를 들면, 기계 판독가능 캐리어 상에 저장될 수 있다.

[0343] 다른 실시 예들은 기계 판독가능 캐리어 상에 저장되는, 여기에 설명된 방법들 중 어느 하나를 실행하기 위한 컴퓨터 프로그램을 포함한다.

[0344] 바꾸어 말하면, 본 발명의 방법의 일 실시 예는 따라서 컴퓨터 프로그램이 컴퓨터 상에 구동할 때, 여기에 설명된 방법들 중 어느 하나를 실행하기 위한 프로그램 코드를 갖는 컴퓨터 프로그램이다.

[0345] 본 발명의 방법의 또 다른 실시 예는 따라서 여기에 설명된 방법들 중 어느 하나를 실행하기 위한 컴퓨터 프로그램을 포함하는, 그 안에 기록되는 데이터 캐리어(또는 데이터 저장 매체,또는 컴퓨터 판독가능 매체)이다. 데이터 캐리어, 디지털 저장 매체 또는 기록 매체는 일반적으로 유형(tangible) 및/또는 비-전이형이다.

[0346] 본 발명의 방법의 또 다른 실시 예는 따라서 여기에 설명된 방법들 중 어느 하나를 실행하기 위한 컴퓨터 프로그램을 나타내는 데이터 스트림 또는 신호들의 시퀀스이다. 데이터 스트림 또는 신호들의 시퀀스는 예를 들면 데이터 통신 연결, 예를 들면 인터넷을 거쳐 전송되도록 구성될 수 있다.

[0347] 또 다른 실시 예는 여기에 설명된 방법들 중 어느 하나를 실행하도록 구성되거나 혹은 적용되는, 처리 수단, 예를 들면 컴퓨터, 또는 프로그램가능 논리 장치를 포함한다.

[0348] 또 다른 실시 예는 그 안에 여기에 설명된 방법들 중 어느 하나를 실행하기 위한 컴퓨터 프로그램이 설치된 컴퓨터를 포함한다.

[0349] 본 발명에 따른 또 다른 실시 예는 여기에 설명된 방법들 중 어느 하나를 실행하기 위한 컴퓨터 프로그램을 수신기로 전송하도록(예를 들면, 전자적으로 또는 선택적으로) 구성되는 장치 또는 시스템을 포함한다. 수신기는 예를 들면, 컴퓨터, 이동 장치, 메모리 장치 등일 수 있다. 장치 또는 시스템은 예를 들면, 컴퓨터 프로그램을 수신기로 전송하기 위한 파일 서버를 포함한다.

[0350] 일부 실시 예들에서, 여기에 설명된 방법들 중 일부 또는 모두를 실행하기 위하여 프로그램가능 논리 장치(예를 들면, 필드 프로그램가능 게이트 어레이)가 사용될 수 있다. 일부 실시 예들에서, 필드 프로그램가능 게이트 어레이는 여기에 설명된 방법들 중 어느 하나를 실행하기 위하여 마이크로프로세서와 협력할 수 있다. 일반적으로, 방법들은 바람직하게는 어떠한 하드웨어 장치에 의해 실행된다.

[0351] 이에 설명된 실시 예들은 단지 본 발명의 원리들을 위한 설명이다. 여기에 설명된 배치들과 상세내용들의 변형

과 변경은 통상의 지식을 가진 자들에 자명할 것이라는 것을 이해할 것이다. 따라서, 본 발명은 여기에 설명된 실시 예들의 설명에 의해 표현된 특정 상세내용이 아닌 특허 청구항의 범위에 의해서만 한정되는 것으로 의도된다.

[0353] 참고문헌

[0354] [1] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, Ajay Luthra, Overview of the H.264/AVC Video Coding Standard, IEEE Trans. Circuits Syst. Video Technol., vol. 13, N7, July 2003.

[0355] [2] JCT-VC, "High-Efficiency Video Coding (HEVC) text specification Working Draft 6", JCTVC-H1003, February 2012.

[0356] [3] ISO/IEC 13818-1: MPEG-2 Systems specification.

부호의 설명

[0358] 5 : 디코더

10 : 화상

12 : 데이터 스트림

14 : 슬라이스

14_1, 14_2, 14_3 및 14_4 : 슬라이스 세그먼트

16 : 슬라이스 순서

18 : 구문 요소 부

20 : 첫 번째 방식

22 : 두 번째 방식

23 : 공간적 이웃 블록

24 : 엔트로피 디코더

26 : 탈양자화 및 역 변환 모듈

27 : 결합기

28 : 예측기

30 : 코딩 블록

32 : 코드 트리블록

34 :이웃 템플릿

36 : 래스터 스캔

42 : 세 번째 방식

50 : 타일

72 : 감산기

400 : 출발 구문 부

898 : 화상

900 : 규칙적인 슬라이스

902, 904 : 열

906 : 타일

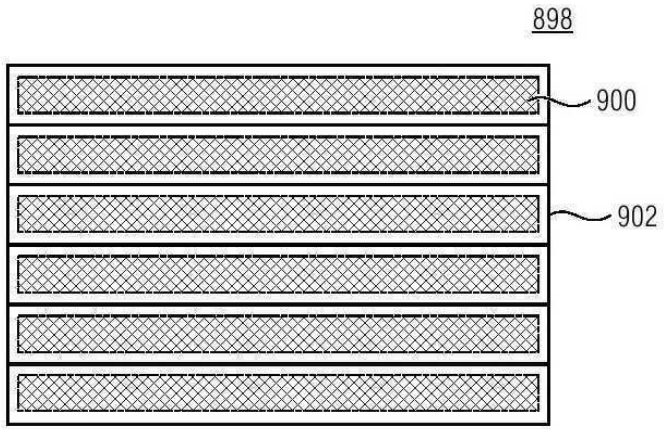
908 : 수평 경계

910 : 수직 경계

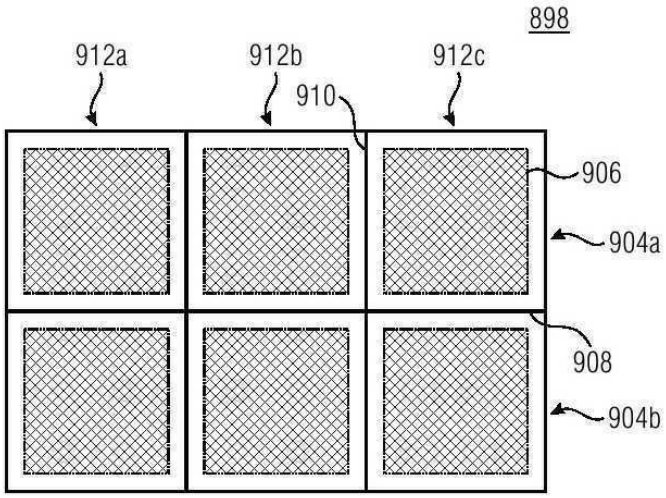
920 : 종속 슬라이스

도면

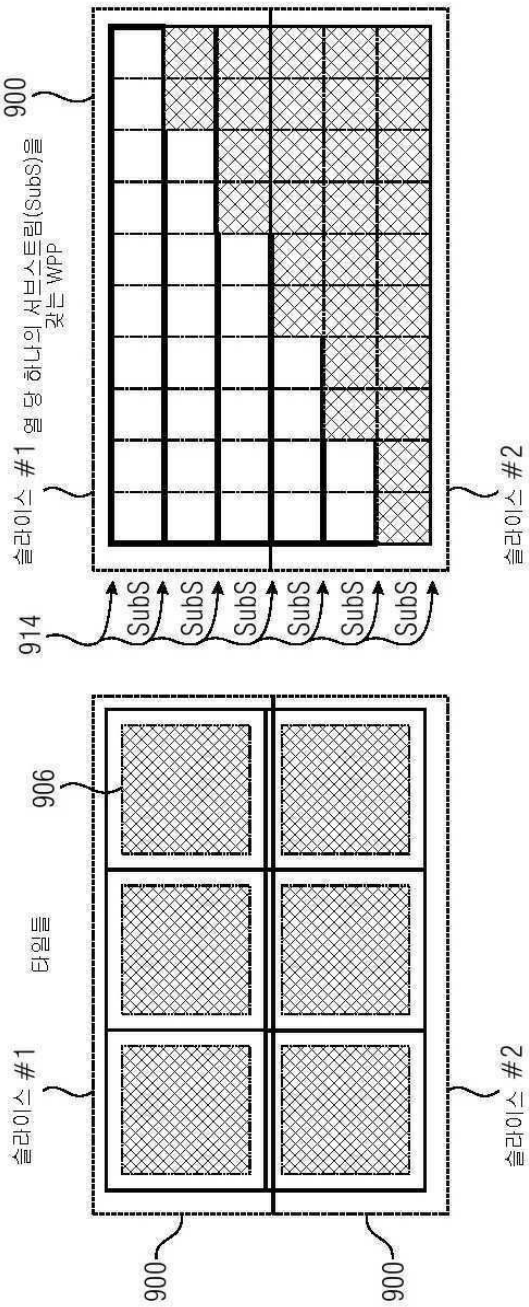
도면1



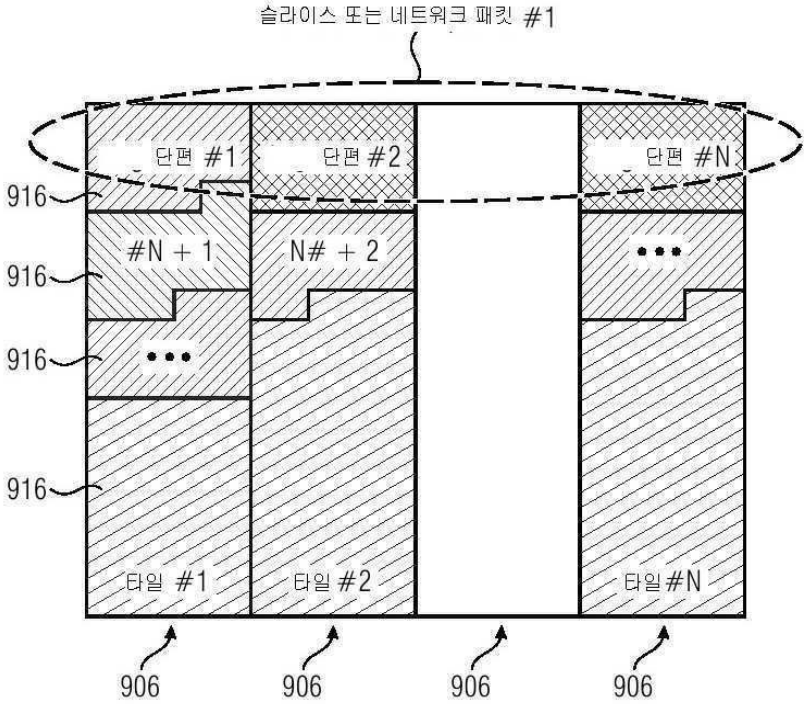
도면2



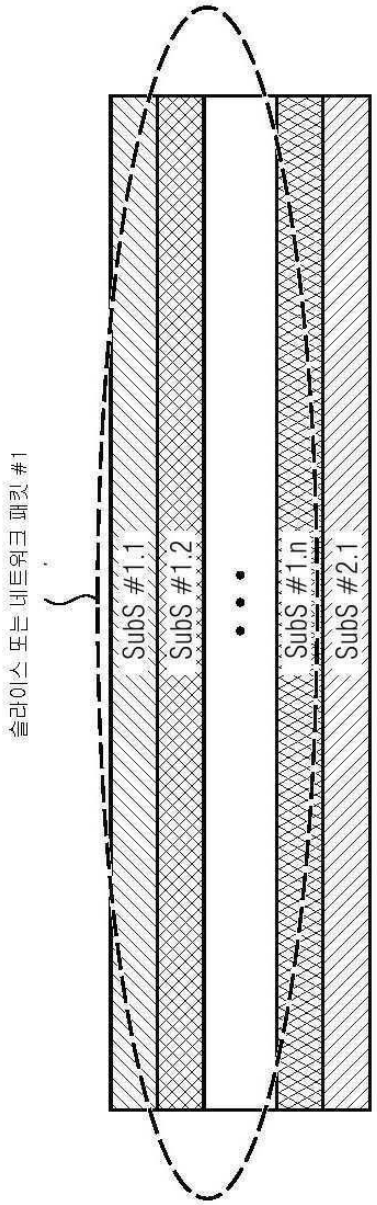
도면3


900
슬라이스 #1
열 당 하나의 서브스트림(SubS)을 갖는 WPP
914
SubS
SubS
SubS
SubS
SubS
SubS
슬라이스 #2
906
타일들
슬라이스 #1
슬라이스 #2
900
900

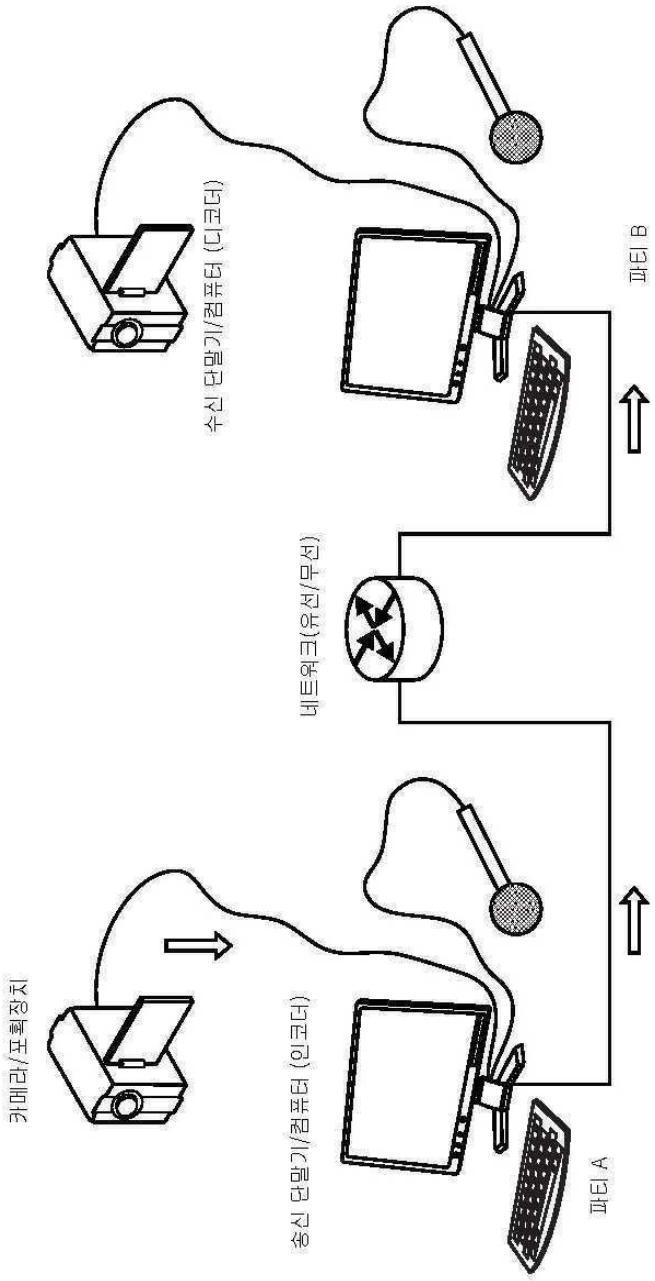
도면4


슬라이스 또는 네트워크 패킷 #1
단편 #1
단편 #2
단편 #N
916
916
916
916
#N + 1
N# + 2
타일 #1
타일 #2
타일#N
906
906
906
906

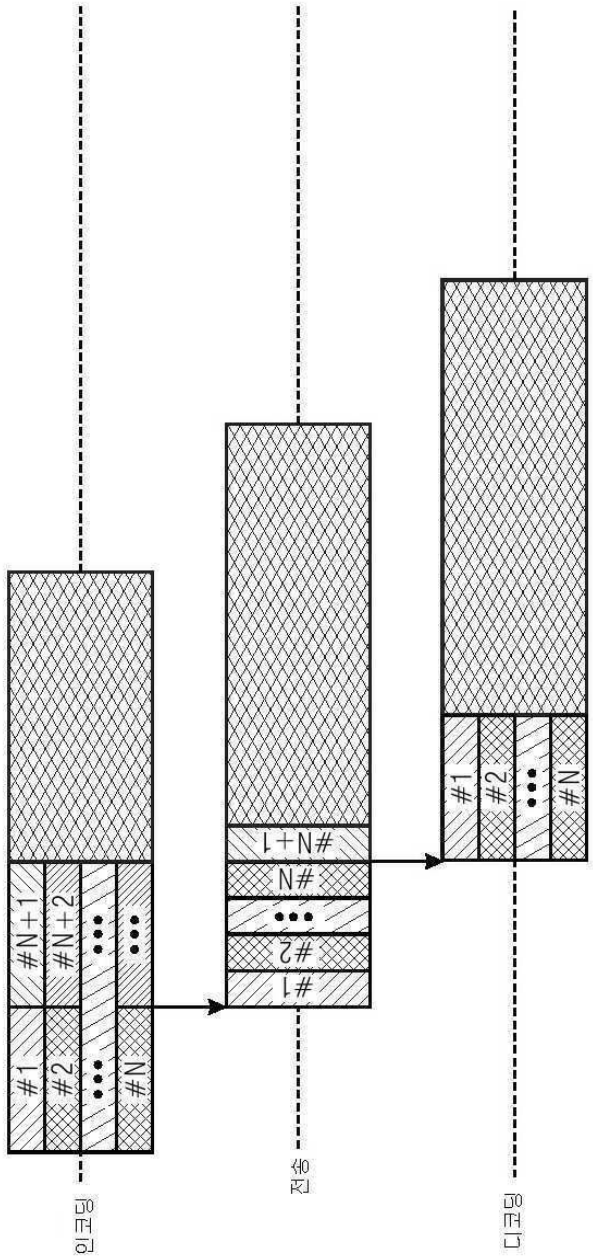
도면5



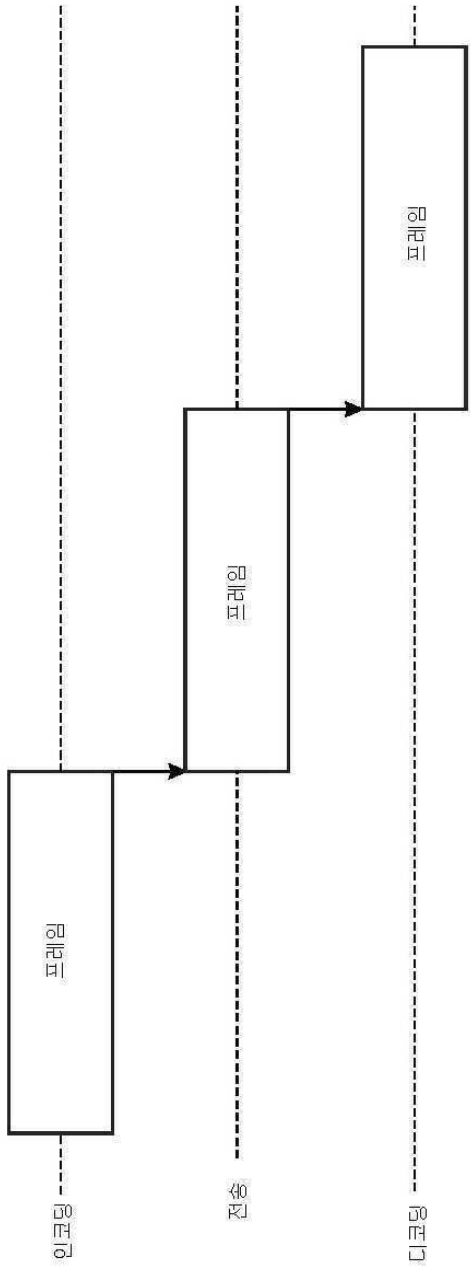
도면6



도면7



도면8

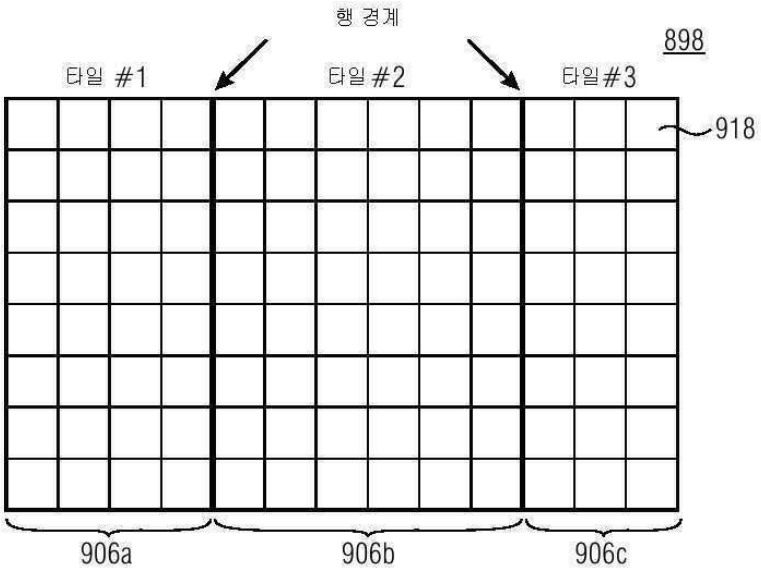


도면9



H.264(09)_F6-7

도면10



도면11a

seq_parameter_set_rbsp( ) {	기술자
profile_idc	u(8)
reserved_zero_8bits /* equal to 0 */	u(8)
level_idc	u(8)
seq_parameter_set_id	ue(v)
chroma_format_idc	ue(v)
if(chroma_format_idc = = 3)	
separate_colour_plane_flag	u(1)
max_temporal_layers_minus1	u(3)
pic_width_in_luma_samples	ue(v)
pic_height_in_luma_samples	ue(v)
pic_cropping_flag	u(1)
if(pic_cropping_flag) {	
pic_crop_left_offset	ue(v)
pic_crop_right_offset	ue(v)
pic_crop_top_offset	ue(v)
pic_crop_bottom_offset	ue(v)
}	
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
pcm_enabled_flag	u(1)
if(pcm_enabled_flag) {	
pcm_bit_depth_luma_minus1	u(4)
pcm_bit_depth_chroma_minus1	u(4)
}	
qpprime_y_zero_transquant_bypass_flag	u(1)
log2_max_pic_order_cnt_lsb_minus4	ue(v)
for(i = 0; i <= max_temporal_layers_minus1; i++) {	
max_dec_pic_buffering[i]	ue(v)

도면11b

num_reorder_pics[i]	ue(v)
max_latency_increase[i]	ue(v)
}	
restricted_ref_pic_lists_flag	u(1)
if(restricted_ref_pic_lists_flag)	
lists_modification_present_flag	u(1)
log2_min_coding_block_size_minus3	ue(v)
log2_diff_max_min_coding_block_size	ue(v)
log2_min_transform_block_size_minus2	ue(v)
log2_diff_max_min_transform_block_size	ue(v)
if(pcm_enabled_flag) {	
log2_min_pcm_coding_block_size_minus3	ue(v)
log2_diff_max_min_pcm_coding_block_size	ue(v)
}	
max_transform_hierarchy_depth_inter	ue(v)
max_transform_hierarchy_depth_intra	ue(v)
scaling_list_enable_flag	
chroma_pred_from_luma_enabled_flag	u(1)
deblocking_filter_in_aps_enabled_flag	u(1)
seq_loop_filter_across_slices_enabled_flag	u(1)
asymmetric_motion_partitions_enabled_flag	u(1)
non_square_quadtree_enabled_flag	u(1)
sample_adaptive_offset_enabled_flag	u(1)
adaptive_loop_filter_enabled_flag	u(1)
if(adaptive_loop_filter_enabled_flag)	
alf_coef_in_slice_flag	u(1)
if(pcm_enabled_flag)	
pcm_loop_filter_disable_flag	u(1)
temporal_id_nesting_flag	u(1)

도면11c

if(log2_min_coding_block_size_minus3 = = 0)	
inter_4x4_enabled_flag	u(1)
num_short_term_ref_pic_sets	ue(v)
for(i = 0; i < num_short_term_ref_pic_sets; i++)	
short_term_ref_pic_set(i)	
long_term_ref_pics_present_flag	u(1)
tiles_or_entropy_coding_sync_idc	u(2)
if(tiles_or_entropy_coding_sync_idc = = 1) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height[i]	ue(v)
}	
loop_filter_across_tiles_enabled_flag	u(1)
}	
vui_parameters_present_flag	u(1)
if(vui_parameters_present_flag)	
vui_parameters( )	
sps_extension_flag	u(1)
if(sps_extension_flag)	
while(more_rbsp_data( ))	
sps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

도면12a

pic_parameter_set_rbsp( ) {	기술자
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
sign_data_hiding_flag	u(1)
if(sign_data_hiding _flag)	
sign_hiding_threshold	u(4)
cabac_init_present_flag	u(1)
num_ref_idx_l0_default_active_minus1	ue(v)
num_ref_idx_l1_default_active_minus1	ue(v)
pic_init_qp_minus26	se(v)
constrained_intra_pred_flag	u(1)
enable_temporal_mvp_flag	u(1)
slice_granularity	u(2)
max_cu_qp_delta_depth	ue(v)
cb_qp_offset	se(v)
cr_qp_offset	se(v)
weighted_pred_flag	u(1)
weighted_bipred_idc	u(2)
output_flag_present_flag	u(1)
if(tiles_or_entropy_coding_sync_idc == 1) {	
tile_info_present_flag	u(1)
tile_control_present_flag	u(1)
if(tile_info_present_flag) {	
num_tile_columns_minus1	ue(v)

도면12b

num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height[i]	ue(v)
}	
}	
if(tile_control_present_flag)	
loop_filter_across_tiles_enabled_flag	u(1)
} else if(tiles_or_entropy_coding_sync_idc == 2)	
num_substreams_minus1	ue(v)
deblocking_filter_control_present_flag	u(1)
if(slice_type == P || slice_type == B)	
log2_parallel_merge_level_minus2	ue(v)
pps_extension_flag	u(1)
if(pps_extension_flag)	
while(more_rbsp_data( ))	
pps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

도면13a

slice_header( ) {	기술자
first_slice_in_pic_flag	u(1)
if(first_slice_in_pic_flag = = 0)	
slice_address	u(v)
slice_type	ue(v)
entropy_slice_flag	u(1)
if(!entropy_slice_flag) {	
pic_parameter_set_id	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(separate_colour_plane_flag = = 1)	
colour_plane_id	u(2)
if(IdrPicFlag) {	
idr_pic_id	ue(v)
no_output_of_prior_pics_flag	u(1)
} else {	
pic_order_cnt_lsb	u(v)
short_term_ref_pic_set_sps_flag	u(1)
if(!short_term_ref_pic_set_sps_flag)	
short_term_ref_pic_set(num_short_term_ref_pic_sets)	
else	
short_term_ref_pic_set_idx	u(v)
if(long_term_ref_pics_present_flag) {	
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_pics; i++) {	
delta_poc_lsb_lt[i]	ue(v)
delta_poc_msb_present_flag[i]	u(1)
if(delta_poc_msb_present_flag[i])	
delta_poc_msb_cycle_lt_minus1[i]	ue(v)
used_by_curr_pic_lt_flag[i]	u(1)
}	
}	
}	
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_interleaving_flag	u(1)
slice_sample_adaptive_offset_flag	u(1)

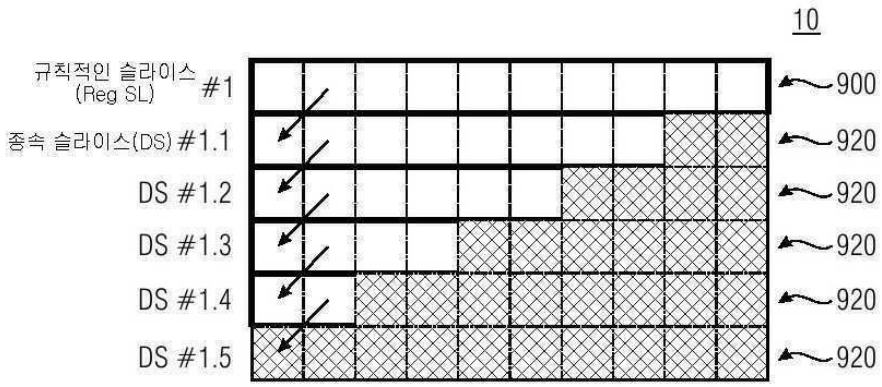
도면13b

if(slice_sao_interleaving_flag & &	
slice_sample_adaptive_offset_flag) {	
sao_cb_enable_flag	u(1)
sao_cr_enable_flag	u(1)
}	
}	
if(scaling_list_enable_flag | |	
deblocking_filter_in_aps_enabled_flag | |	
(sample_adaptive_offset_enabled_flag) & & !slice_sao_interleaving_flag) | |	
adaptive_loop_filter_enabled_flag)	
aps_id	ue(v)
if(slice_type = = P | | slice_type = = B) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type = = B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
}	
if(lists_modification_present_flag) {	
ref_pic_list_modification( )	
ref_pic_list_combination( )	
}	
if(slice_type = = B)	
mvd_l1_zero_flag	u(1)
}	
if(cabac_init_present_flag && slice_type != I)	
cabac_init_flag	u(1)
if(!entropy_slice_flag) {	
slice_qp_delta	se(v)
if(deblocking_filter_control_present_flag) {	
if(deblocking_filter_in_aps_enabled_flag)	
inherit_dbl_params_from_aps_flag	u(1)
if(!inherit_dbl_params_from_aps_flag) {	
disable_deblocking_filter_flag	u(1)
if(!disable_deblocking_filter_flag) {	
beta_offset_div2	se(v)

도면13c

tc_offset_div2	se(v)
}	
}	
}	
if(slice_type = = B)	
collocated_from_l0_flag	u(1)
if(slice_type != I && ((collocated_from_l0_flag && num_ref_idx_l0_active_minus1 > 0) || (!collocated_from_l0_flag && num_ref_idx_l1_active_minus1 > 0))	
collocated_ref_idx	ue(v)
if((weighted_pred_flag && slice_type = = P) || (weighted_bipred_idc = = 1 && slice_type = = B))	
pred_weight_table( )	
}	
if(slice_type = = P || slice_type = = B)	
five_minus_max_num_merge_cand	ue(v)
if(adaptive_loop_filter_enabled_flag) {	
slice_adaptive_loop_filter_flag	u(1)
if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
alf_param( )	
if(slice_adaptive_loop_filter_flag && !alf_coef_in_slice_flag)	
alf_cu_control_param( )	
}	
if(seq_loop_filter_across_slices_enabled_flag && (slice_adaptive_loop_filter_flag || slice_sample_adaptive_offset_flag || !disable_deblocking_filter_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
if(tiles_or_entropy_coding_sync_idc > 0) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset[i]	u(v)
}	
}	
}	

도면14



도면15

pic_parameter_set_rbsp( ) {	기술자
• • •	
dependent_slices_present_flag	u(1)
last_ctb_cabac_init_flg	u(1)
}	

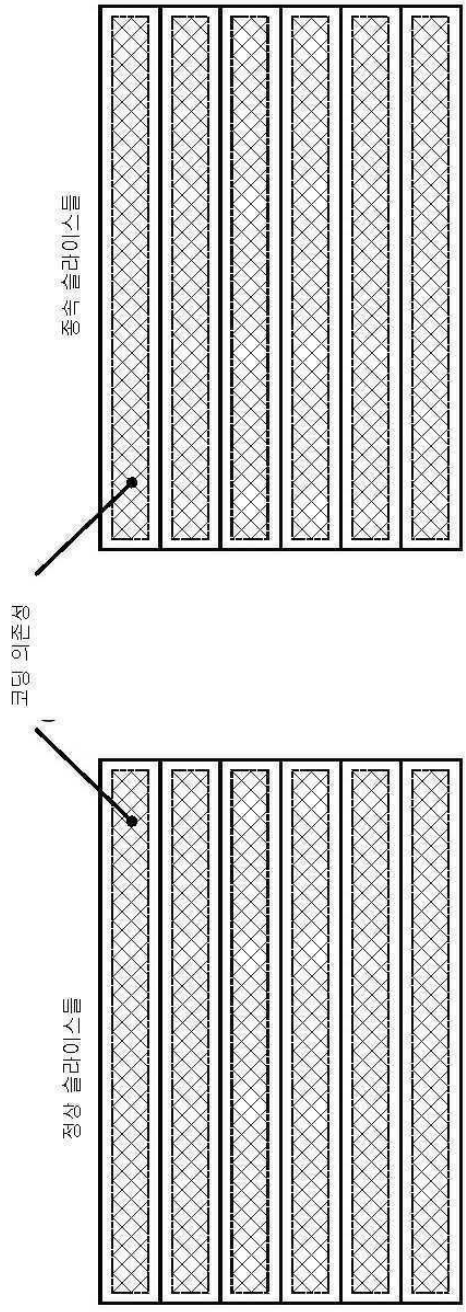
도면16

400

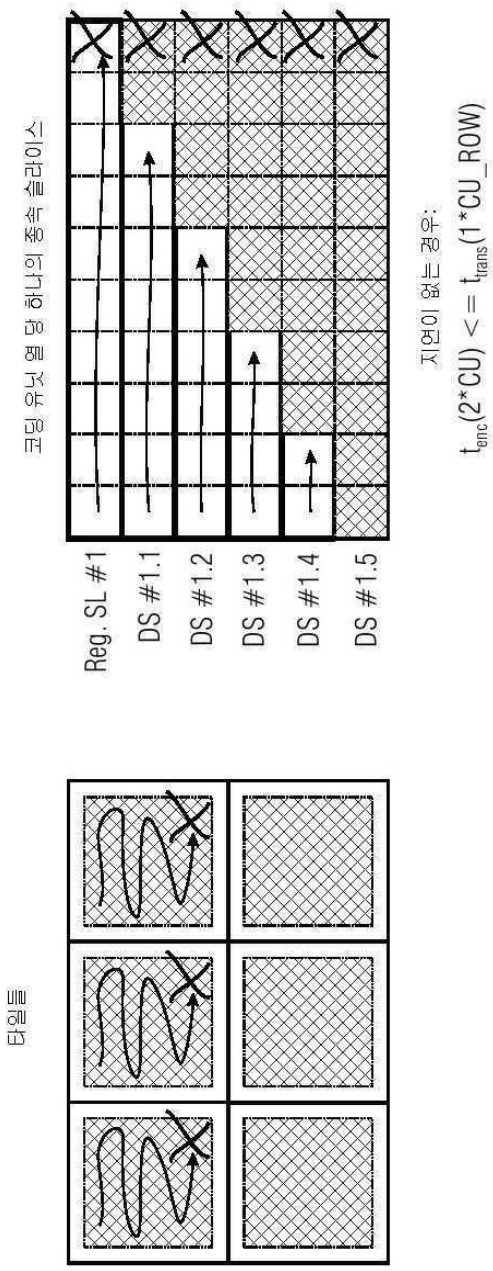
100

slice_header( ) {	기술자
first_slice_in_pic_flag	u(1)
if(first_slice_in_pic_flag = = 0)	
slice_address	u(v)
slice_type	ue(v)
entropy_slice_flag	u(1)
dependent_slice_flag	u(1)
if(!entropy_slice_flag) {	
if(!dependent_slice_flag) {	
• • •	
} else {	
no_cabac_reset_flag	u(1)
}	
• • •	
if(cabac_init_present_flag && slice_type != I && !no_cabac_reset_flag)	
cabac_init_flag	u(1)
• • •	
if(slice_type = = P | | slice_type = = B)	
five_minus_max_num_merge_cand	ue(v)
if(adaptive_loop_filter_enabled_flag) {	
slice_adaptive_loop_filter_flag	u(1)
if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
alf_param( )	
if(slice_adaptive_loop_filter_flag && !alf_coef_in_slice_flag)	
alf_cu_control_param( )	
}	
if(seq_loop_filter_across_slices_enabled_flag && (slice_adaptive_loop_filter_flag | | slice_sample_adaptive_offset_flag | | !disable_deblocking_filter_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
if (tiles_or_entropy_coding_sync_idc > = 0) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset[i]	u(v)
}	
}	
}	

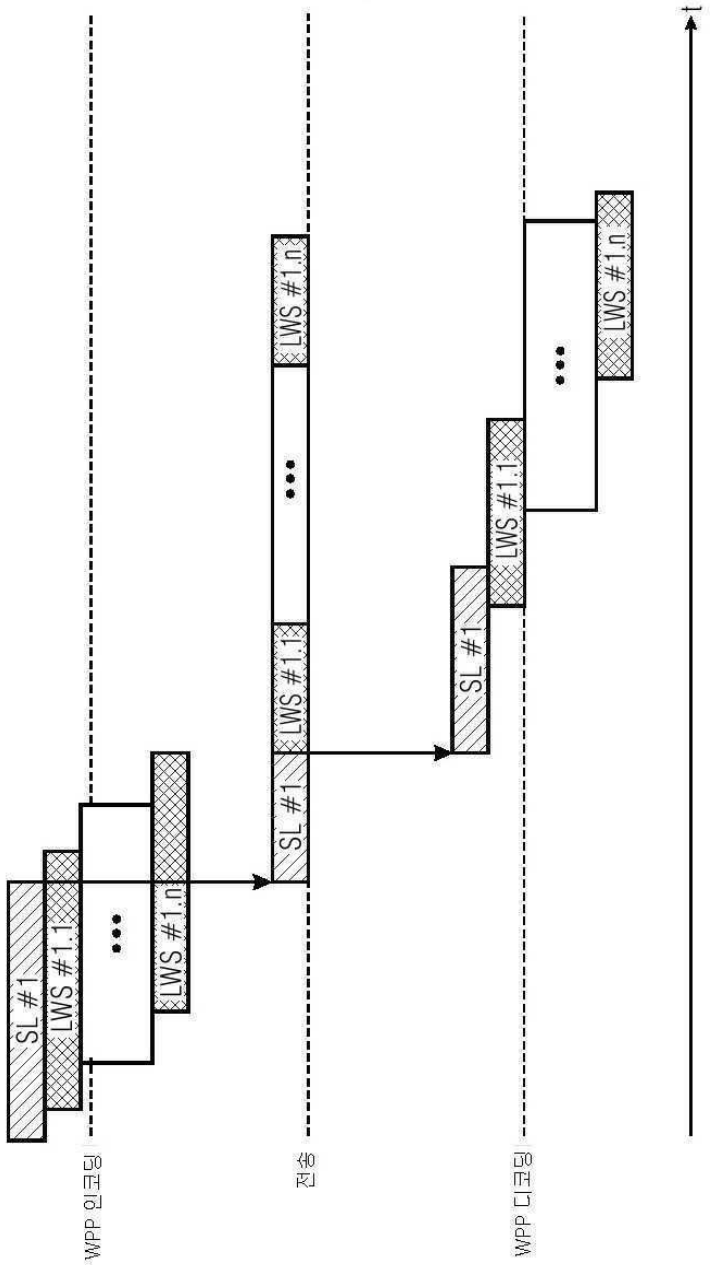
도면17



도면18


타일들
코딩 유닛 열 당 하나의 종속 슬라이스
Reg. SL #1
DS #1.1
DS #1.2
DS #1.3
DS #1.4
DS #1.5
지연이 없는 경우:
$t_{enc}(2*CU) <= t_{trans}(1*CU_ROW)$

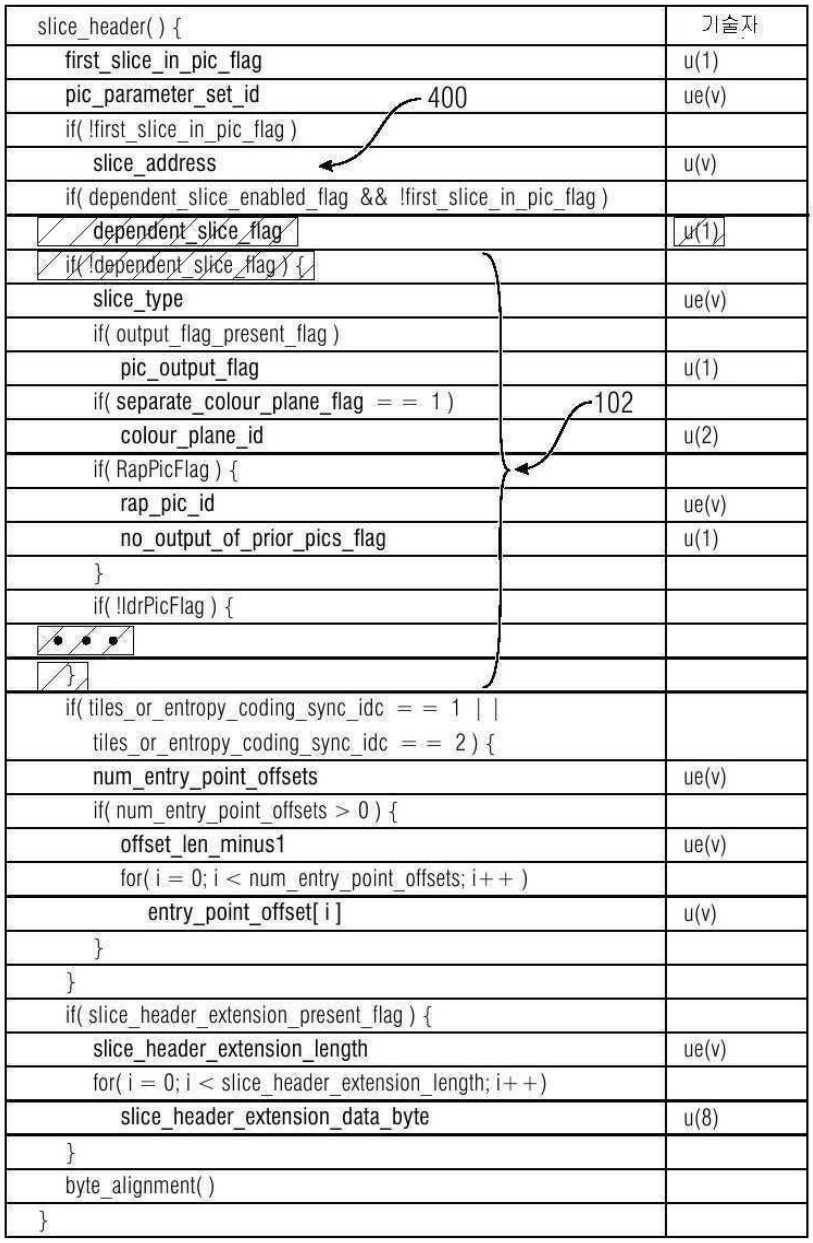
도면19


SL #1
LWS #1.1
LWS #1.n
SL #1
LWS #1.1
LWS #1.n
SL #1
LWS #1.1
LWS #1.n
t
WPP 인코딩
전송
WPP 디코딩

도면20


종속 슬라이스(DS)들을 갖는 규칙적인 슬라이스(RS)들
RS
DS
DS
RS
DS
DS

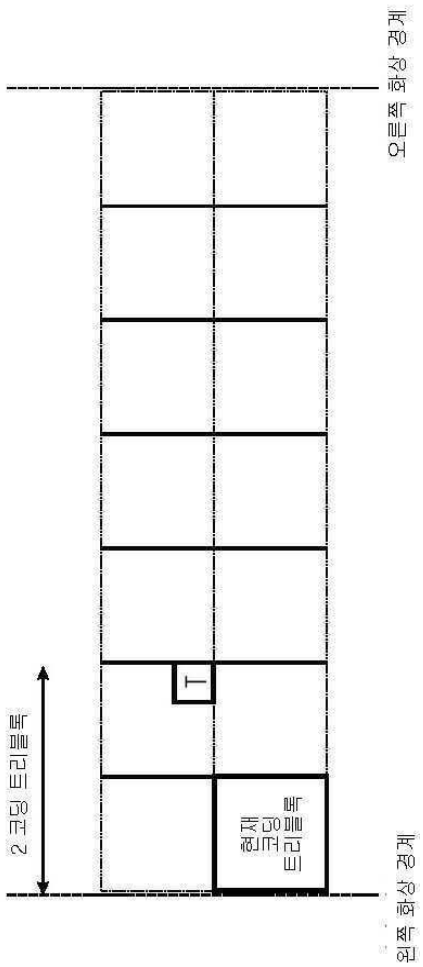
도면21


slice_header( ) {
기술자
first_slice_in_pic_flag
u(1)
pic_parameter_set_id
ue(v)
400
if(!first_slice_in_pic_flag)
slice_address
u(v)
if(dependent_slice_enabled_flag && !first_slice_in_pic_flag)
dependent_slice_flag
u(1)
if(!dependent_slice_flag) {
slice_type
ue(v)
if(output_flag_present_flag)
pic_output_flag
u(1)
if(separate_colour_plane_flag == 1)
102
colour_plane_id
u(2)
if(RapPicFlag) {
rap_pic_id
ue(v)
no_output_of_prior_pics_flag
u(1)
}
if(!IdrPicFlag) {
• • •
}
if(tiles_or_entropy_coding_sync_idc == 1 | |
tiles_or_entropy_coding_sync_idc == 2) {
num_entry_point_offsets
ue(v)
if(num_entry_point_offsets > 0) {
offset_len_minus1
ue(v)
for(i = 0; i < num_entry_point_offsets; i++)
entry_point_offset[i]
u(v)
}
}
if(slice_header_extension_present_flag) {
slice_header_extension_length
ue(v)
for(i = 0; i < slice_header_extension_length; i++)
slice_header_extension_data_byte
u(8)
}
byte_alignment( )
}

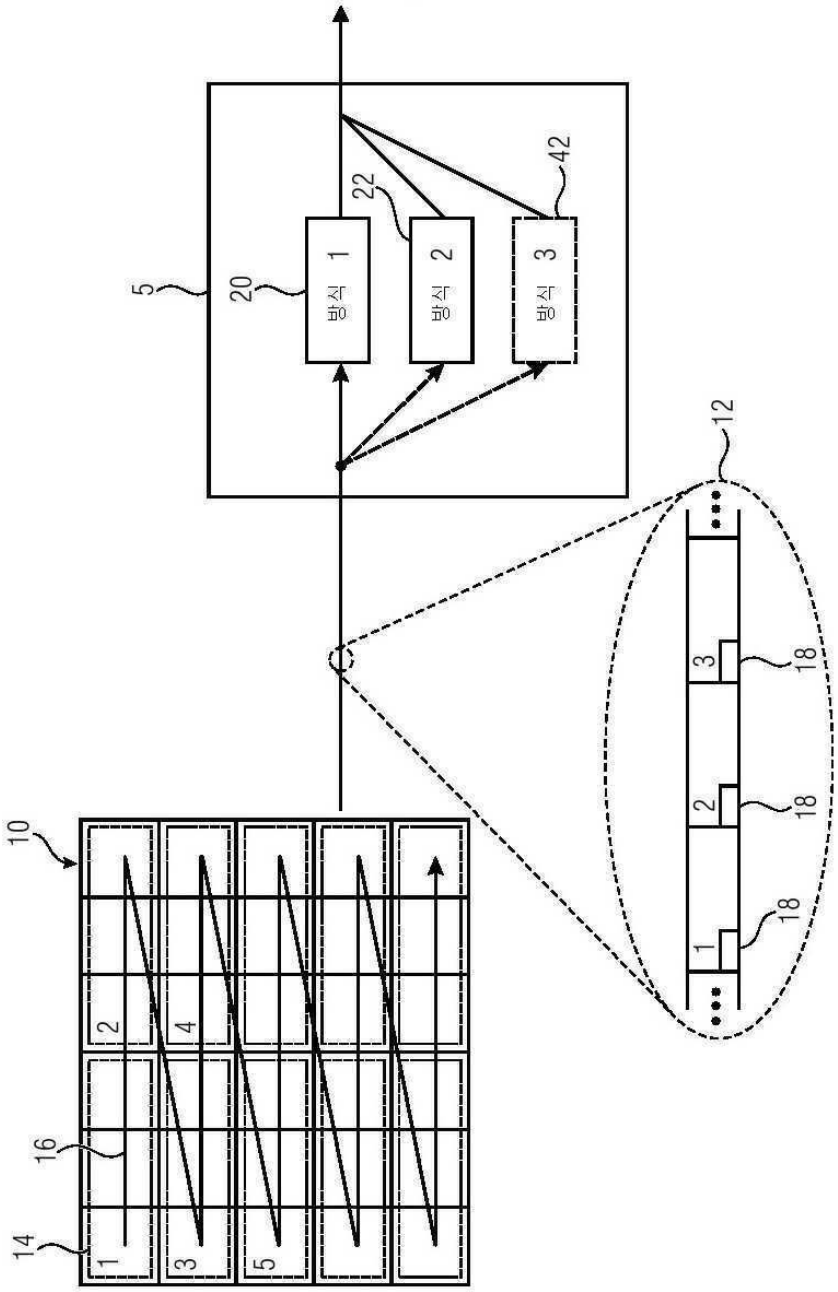
도면22

pic_parameter_set_rbsp( ) {	기술자
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
sign_data_hiding_flag	u(1)
if(sign_data_hiding _flag)	
sign_hiding_threshold	u(4)
cabac_init_present_flag	u(1)
num_ref_idx_l0_default_active_minus1	ue(v)
num_ref_idx_l1_default_active_minus1	ue(v)
[Ed. (BB): not present in HM software]	
pic_init_qp_minus26	se(v)
constrained_intra_pred_flag	u(1)
slice_granularity	u(2)
diff_cu_qp_delta_depth	ue(v)
cb_qp_offset	se(v)
cr_qp_offset	se(v)
weighted_pred_flag	u(1)
weighted_bipred_idc	u(2)
output_flag_present_flag	u(1)
transquant_bypass_enable_flag	u(1)
dependent_slice_enabled_flag	u(1)
tiles_or_entropy_coding_sync_idc	u(2)
if(tiles_or_entropy_coding_sync_idc == 1) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height[i]	ue(v)
}	
loop_filter_across_tiles_enabled_flag	u(1)
} else if(tiles_or_entropy_coding_sync_idc == 3)	
cabac_independent_flag	u(1)
• • •	
}	

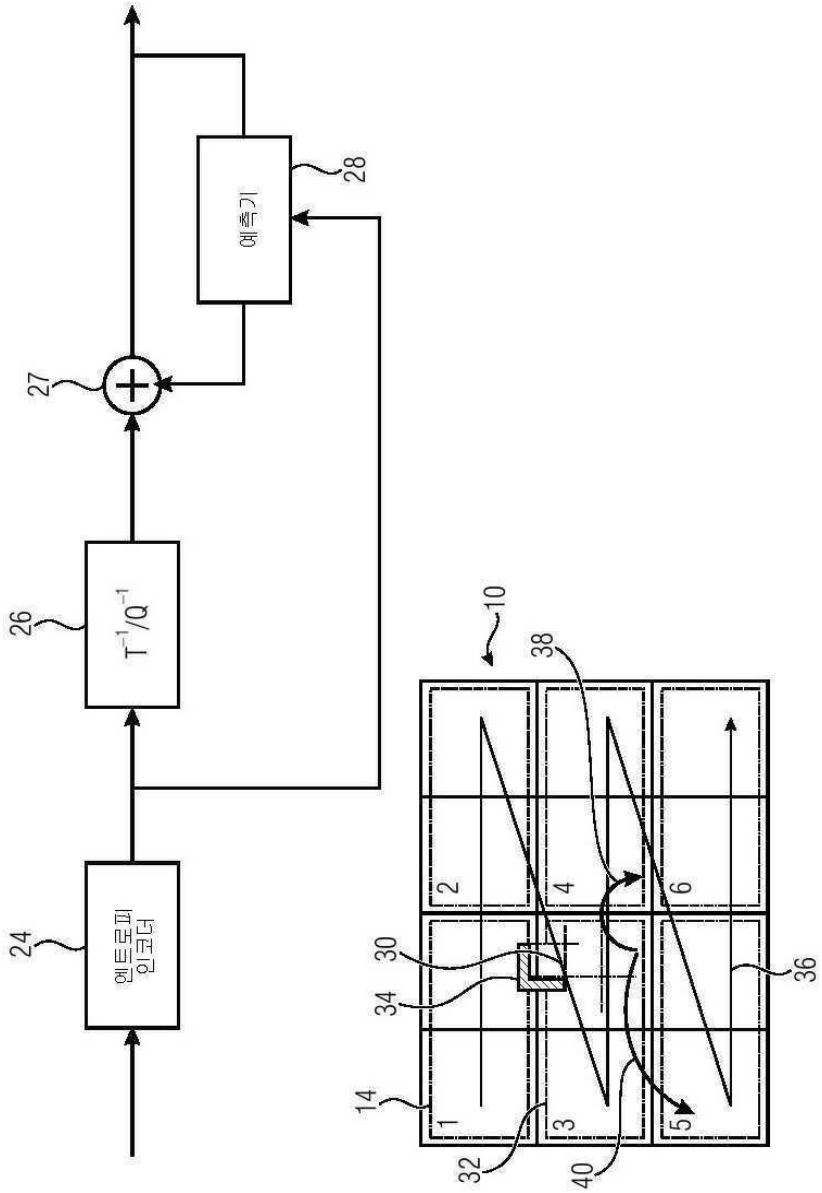
도면23



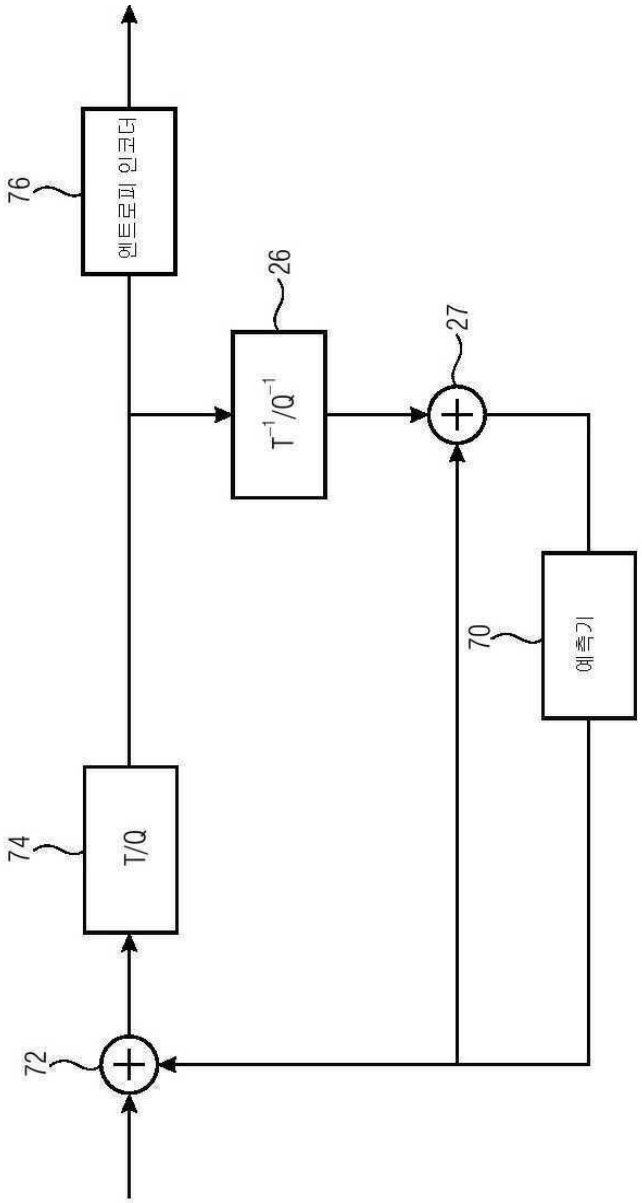
도면24



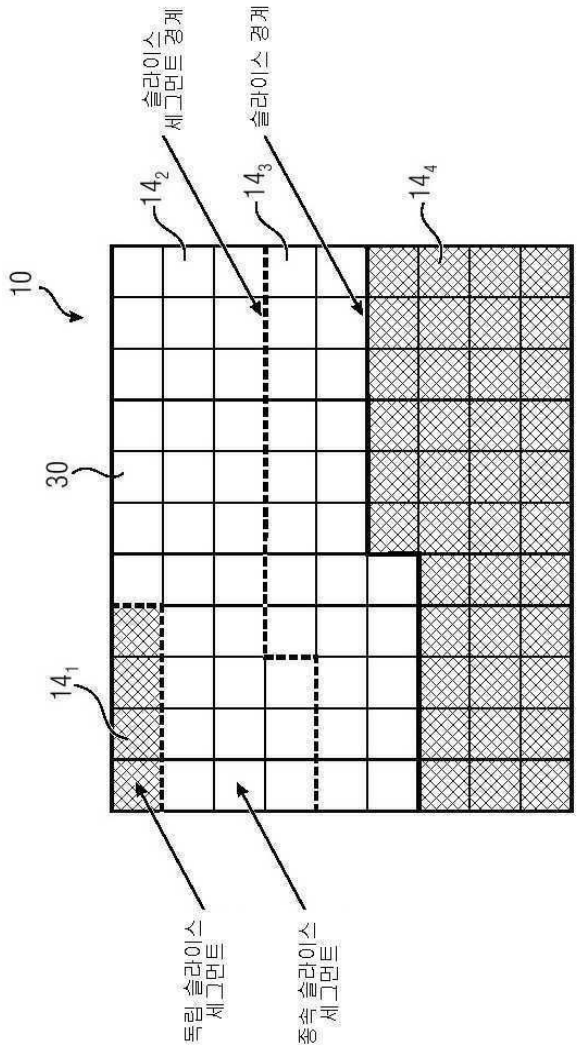
도면25



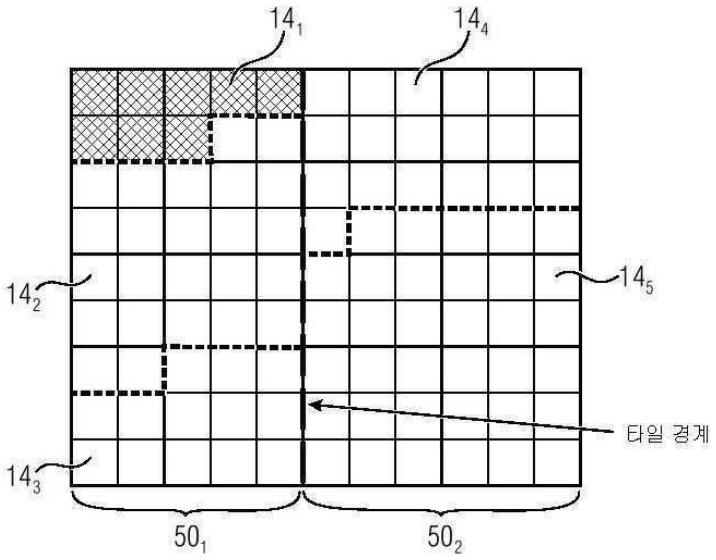
도면26



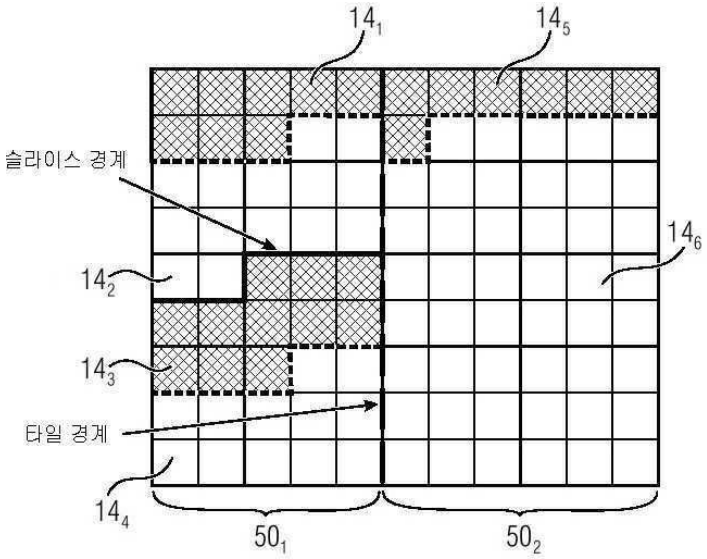
도면27


슬라이스 세그먼트 경계
슬라이스 경계
14_2
14_3
14_4
10
30
14_1
독립 슬라이스 세그먼트
종속 슬라이스 세그먼트

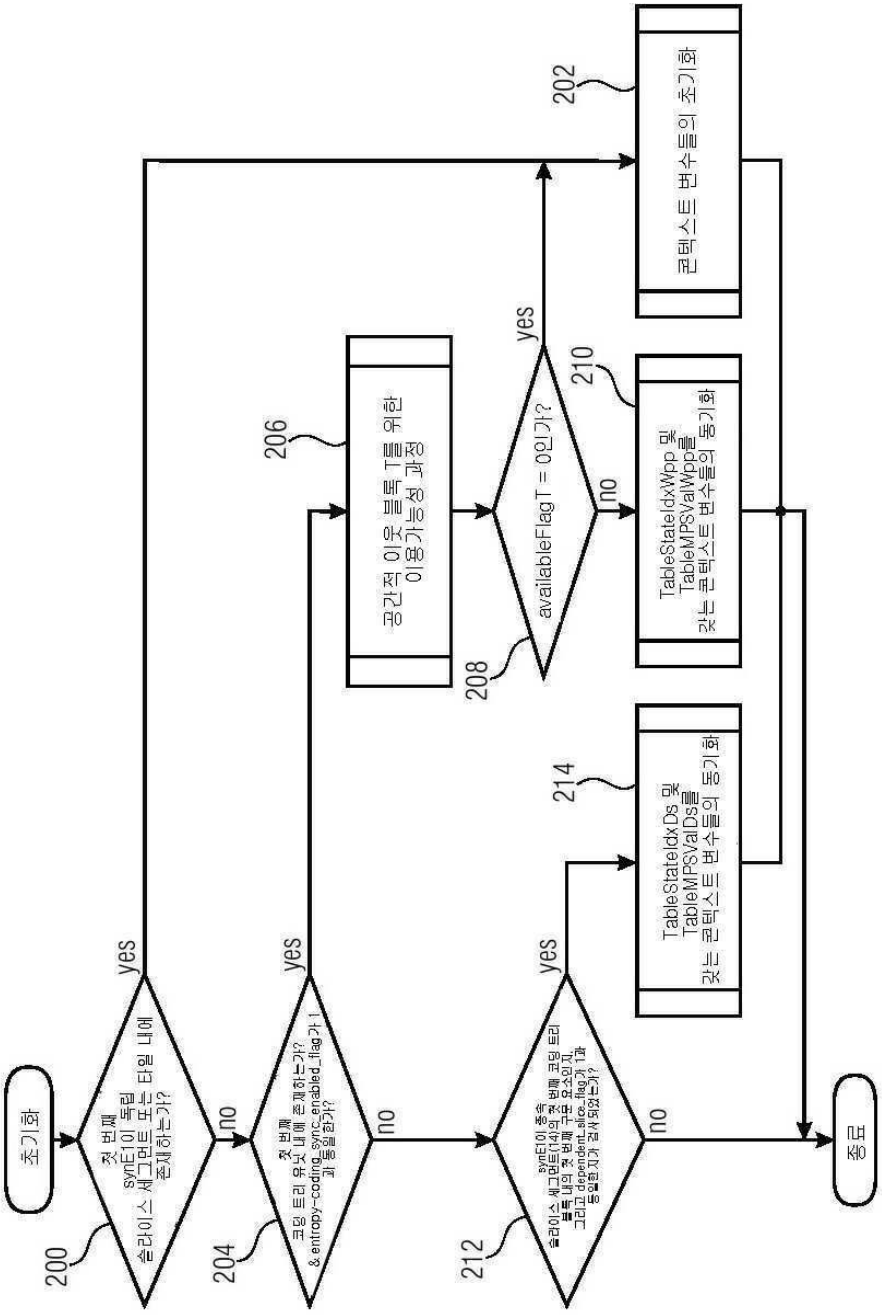
도면28a


14_1
14_4
14_2
14_5
타일 경계
14_3
50_1
50_2

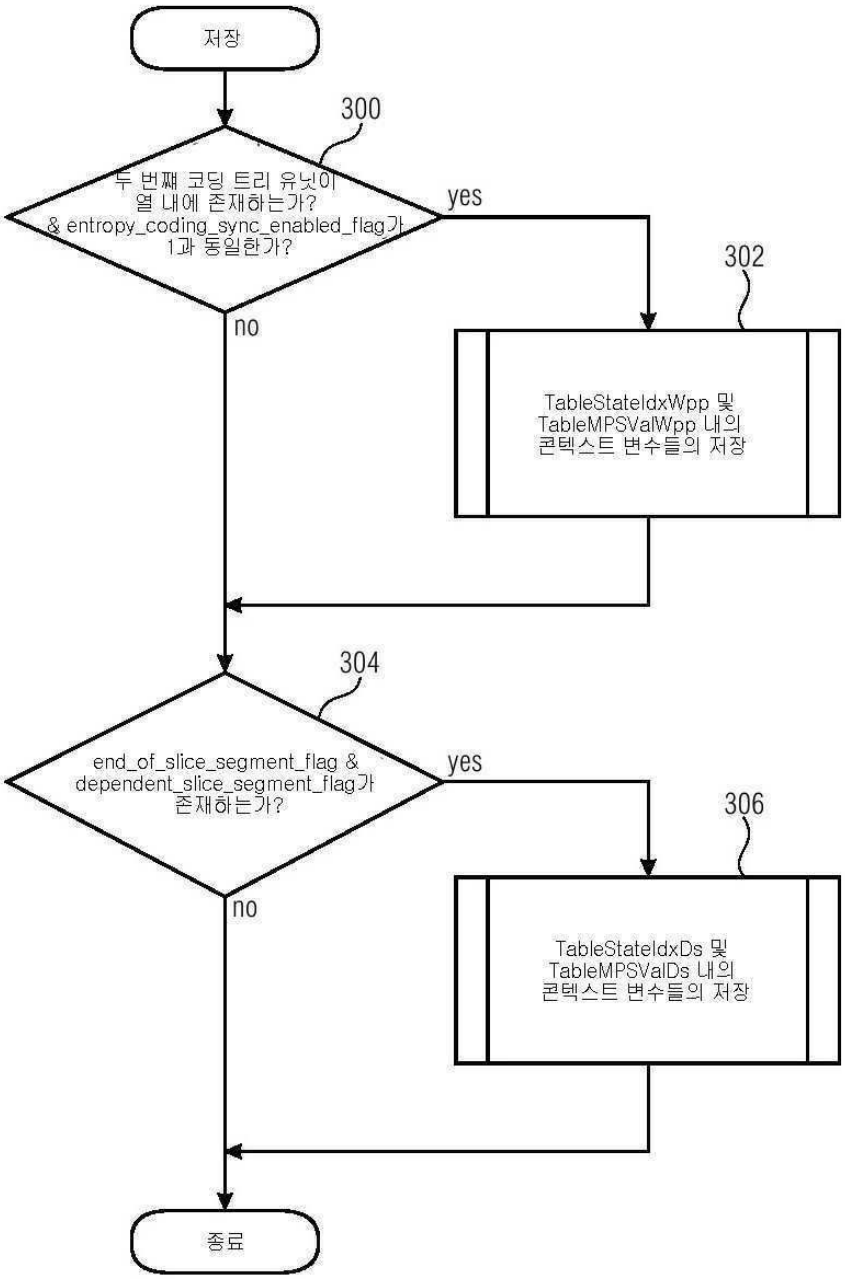
도면28b



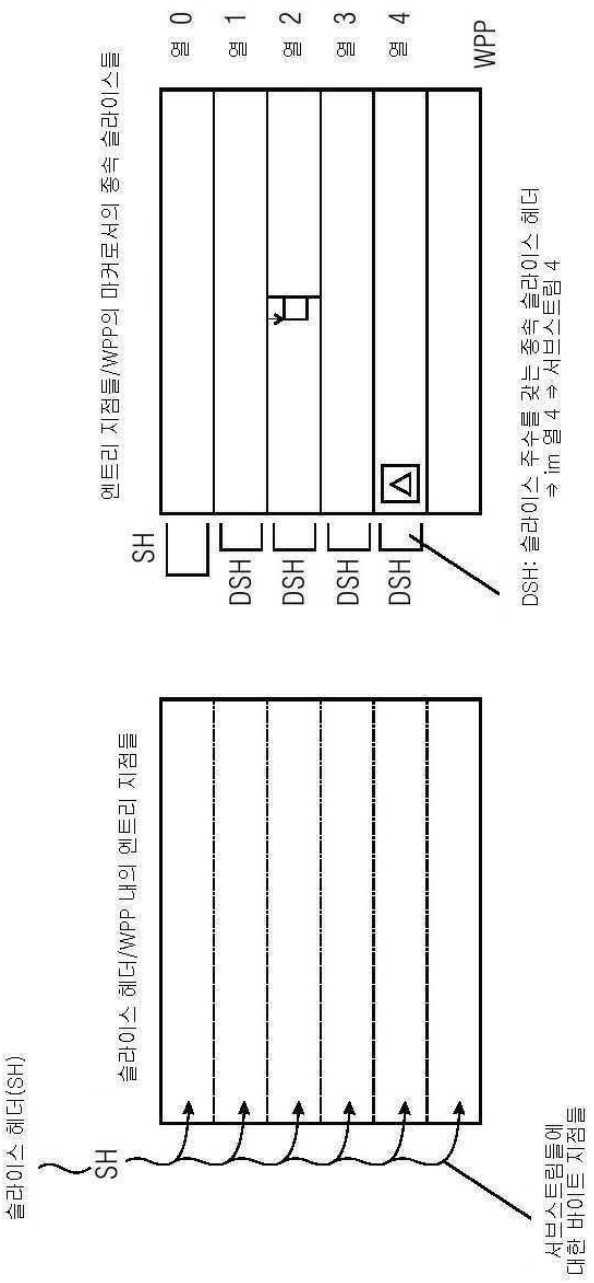
도면29


초기화
첫 번째 sync단이 독립 슬라이스 세그먼트 또는 타일 내에 존재하는가?
yes
no
200
첫 번째 코딩 트리 유닛 내에 존재하는가? & entropy_coding_sync_enabled_flag가 1과 동일한가?
yes
no
204
공간적 이웃 블록 T를 위한 이용가능성 과정
206
availableFlagT = 0인가?
yes
no
208
콘텍스트 변수들의 초기화
202
TableStateIdxWpp 및 TableMPSValWpp를 갖는 콘텍스트 변수들의 동기화
210
TableStateIdxDs 및 TableMPSValDs를 갖는 콘텍스트 변수들의 동기화
214
sync단이 종속 슬라이스 세그먼트(14)의 첫 번째 CTB 블록 내의 첫 번째 구문 요소인지, 그리고 dependent_slice_flag가 1과 동일한지 결정되었는가?
yes
no
212
종료

도면30


저장
300
두 번째 코딩 트리 유닛이
열 내에 존재하는가?
& entropy_coding_sync_enabled_flag가
1과 동일한가?
yes
no
302
TableStateIdxWpp 및
TableMPSValWpp 내의
콘텍스트 변수들의 저장
304
end_of_slice_segment_flag &
dependent_slice_segment_flag가
존재하는가?
yes
no
306
TableStateIdxDs 및
TableMPSValDs 내의
콘텍스트 변수들의 저장
종료

도면31


엔트리 지점들/WPP의 마커로서의 종속 슬라이스들
셀 0
셀 1
셀 2
셀 3
셀 4
WPP
SH
DSH
DSH
DSH
DSH
DSH: 슬라이스 주소를 갖는 종속 슬라이스 헤더
⇒ im 셀 4 ⇒ 서브스트림 4
슬라이스 헤더(SH)
SH
슬라이스 헤더/WPP 내의 엔트리 지점들
서브스트림들에
대한 바이트 지점들