(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2016-0134730
(43) 공개일자 2016년11월23일

(51) 국제특허분류(Int. Cl.)
H04N 19/60 (2014.01) *H04N 19/157* (2014.01)
H04N 19/18 (2014.01) *H04N 19/42* (2014.01)
(52) CPC특허분류
H04N 19/60 (2015.01)
H04N 19/157 (2015.01)
(21) 출원번호 10-2016-7028266
(22) 출원일자(국제) 2015년03월12일
심사청구일자 없음
(85) 번역문제출일자 2016년10월11일
(86) 국제출원번호 PCT/US2015/020237
(87) 국제공개번호 WO 2015/179010
국제공개일자 2015년11월26일
(30) 우선권주장
14/216,369 2014년03월17일 미국(US)

(71) 출원인
퀄컴 인코포레이티드
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
(72) 발명자
정 인석
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
유 양
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
(뒷면에 계속)
(74) 대리인
특허법인코리아나

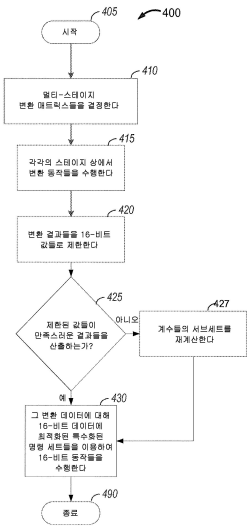
전체 청구항 수 : 총 30 항

(54) 발명의 명칭 **메시-기반의 계산들을 이용한 저 복잡성 순방향 변환들을 위한 시스템들 및 방법들**

(57) 요 약

메시-기반의 계산들을 이용한 저 복잡성 순방향 변환들을 위한 시스템들 및 방법들이 본원에서 설명된다. 본 개시물에서 설명되는 기술요지의 일 양태는 비디오 정보를 저장하도록 구성된 메모리를 포함하는 비디오 인코더를 제공한다. 비디오 인코더는 메모리와 통신하는 프로세서를 더 포함한다. 프로세서는 변환을 다수의 변환 스테이지들로 분해하도록 구성된다. 프로세서는 각각의 변환 스테이지에서의 변환 스테이지 출력을 결정하기 위해 다수의 스테이지들을 이용하여 비디오 정보를 변환하도록 더 구성된다. 프로세서는 각각의 변환 스테이지에서의 변환 스테이지 출력을 미리 결정된 비트 심도로 제한하도록 더 구성된다. 프로세서는 다수의 스테이지들의 최종 스테이지의 제한된 변환 출력에 대한 동작들을 수행하도록 더 구성되며, 동작들은 단지 미리 결정된 비트 심도를 가지는 데이터와 함께 사용하는데 이용가능하다.

대 표 도 - 도4



(52) CPC특허분류
H04N 19/18 (2015.01)
H04N 19/42 (2015.01)
(72) 발명자
왕 샹린
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
카르체비츠 마르타
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775

명 세 서

청구범위

청구항 1

비디오 인코더로서,

비디오 정보를 저장하도록 구성된 메모리; 및

상기 메모리와 통신하는 프로세서를 포함하며,

상기 프로세서는,

변환을 다수의 변환 스테이지들로 분해하고;

각각의 변환 스테이지에서의 변환 스테이지 출력을 결정하기 위해 상기 다수의 스테이지들을 이용하여 상기 비디오 정보를 변환하고;

상기 각각의 변환 스테이지에서의 상기 변환 스테이지 출력을 미리 결정된 비트 심도로 제한하고; 그리고

상기 다수의 스테이지들의 최종 스테이지의 제한된 변환 출력에 대한 동작들을 수행하도록

구성되며,

상기 동작들은 단지 상기 미리 결정된 비트 심도를 가지는 데이터와 함께 사용하기 위해 이용가능한, 비디오 인코더.

청구항 2

제 1 항에 있어서,

상기 프로세서는 메시-기반의 방법, 버터플라이 방법, 또는 짝수-홀수 분해를 이용하여 상기 변환을 분해하도록 더 구성되는, 비디오 인코더.

청구항 3

제 1 항에 있어서,

상기 프로세서는, 클리핑된 상기 변환 스테이지 출력이 반드시 미리 결정된 범위 내에 들어가도록 상기 변환 스테이지 출력을 클리핑함으로써 상기 변환 스테이지 출력을 제한하도록 더 구성되는, 비디오 인코더.

청구항 4

제 1 항에 있어서,

상기 미리 결정된 비트 심도는 16-비트인, 비디오 인코더.

청구항 5

제 1 항에 있어서,

상기 프로세서는 상기 제한된 변환 스테이지 출력과 상기 변환 스테이지 출력 사이의 편차를 결정하도록 더 구성되는, 비디오 인코더.

청구항 6

제 5 항에 있어서,

상기 편차는 상기 제한된 변환 스테이지 출력과 상기 변환 스테이지 출력 사이의 차이를 포함하는, 비디오 인코더.

청구항 7

제 5 항에 있어서,

상기 프로세서는, 상기 편차가 미리 결정된 임계값보다 클 때 계수들의 서브세트를 적어도 하나의 변환 스테이지 내에서 재계산하도록 더 구성되는, 비디오 인코더.

청구항 8

제 7 항에 있어서,

상기 프로세서는 적어도 재-계산된 상기 계수들의 서브세트를 이용하여 상기 비디오 정보를 변환하도록 더 구성되는, 비디오 인코더.

청구항 9

비디오를 인코딩하는 방법으로서,

비디오 정보를 저장하는 단계;

변환을 다수의 변환 스테이지들로 분해하는 단계;

각각의 변환 스테이지에서의 변환 스테이지 출력을 결정하기 위해 상기 다수의 스테이지들을 이용하여 상기 비디오 정보를 변환하는 단계;

상기 각각의 변환 스테이지에서의 상기 변환 스테이지 출력을 미리 결정된 비트 심도로 제한하는 단계; 및

상기 다수의 스테이지들의 최종 스테이지의 제한된 변환 출력에 대한 동작들을 수행하는 단계를 포함하며,

상기 동작들은 단지 상기 미리 결정된 비트 심도를 가지는 데이터와 함께 사용하기 위해 이용가능한, 비디오를 인코딩하는 방법.

청구항 10

제 9 항에 있어서,

메시-기반의 방법, 버터플라이 방법, 또는 짝수-홀수 분해를 이용하여 상기 변환을 분해하는 단계를 더 포함하는, 비디오를 인코딩하는 방법.

청구항 11

제 9 항에 있어서,

클리핑된 상기 변환 스테이지 출력이 반드시 미리 결정된 범위 내에 들어가도록 상기 변환 스테이지 출력을 클리핑함으로써 상기 변환 스테이지 출력을 제한하는 단계를 더 포함하는, 비디오를 인코딩하는 방법.

청구항 12

제 9 항에 있어서,

상기 미리 결정된 비트 심도는 16-비트인, 비디오를 인코딩하는 방법.

청구항 13

제 9 항에 있어서,

상기 제한된 변환 스테이지 출력과 상기 변환 스테이지 출력 사이의 편차를 결정하는 단계를 더 포함하는, 비디오를 인코딩하는 방법.

청구항 14

제 13 항에 있어서,

상기 편차는 상기 제한된 변환 스테이지 출력과 상기 변환 스테이지 출력 사이의 차이를 포함하는, 비디오를 인

코딩하는 방법.

청구항 15

제 13 항에 있어서,

상기 편차가 미리 결정된 임계값보다 클 때 계수들의 서브세트를 적어도 하나의 변환 스테이지 내에서 재계산하는 단계를 더 포함하는, 비디오를 인코딩하는 방법.

청구항 16

제 15 항에 있어서,

적어도 재-계산된 상기 계수들의 서브세트를 이용하여 상기 비디오 정보를 변환하는 단계를 더 포함하는, 비디오를 인코딩하는 방법.

청구항 17

코드를 포함하는 비일시적 컴퓨터-판독가능 매체로서,

상기 코드는, 실행될 경우, 장치로 하여금,

비디오 정보를 저장하게 하고;

변환을 다수의 변환 스테이지들로 분해하게 하고;

각각의 변환 스테이지에서의 변환 스테이지 출력을 결정하기 위해 상기 다수의 스테이지들을 이용하여 상기 비디오 정보를 변환하게 하고;

상기 각각의 변환 스테이지에서의 상기 변환 스테이지 출력을 미리 결정된 비트 심도로 제한하게 하고; 그리고

상기 다수의 스테이지들의 최종 스테이지의 제한된 변환 출력에 대한 동작들을 수행하게 하며,

상기 동작들은 단지 상기 미리 결정된 비트 심도를 가지는 데이터와 함께 사용하기 위해 이용가능한, 비일시적 컴퓨터-판독가능 매체.

청구항 18

제 17 항에 있어서,

실행될 경우, 상기 장치로 하여금, 메시-기반의 방법, 버터플라이 방법, 또는 짝수-홀수 분해를 이용하여 상기 변환을 분해하게 하는 코드를 더 포함하는, 비일시적 컴퓨터-판독가능 매체.

청구항 19

제 17 항에 있어서,

실행될 경우, 상기 장치로 하여금, 클리핑된 상기 변환 스테이지 출력이 반드시 미리 결정된 범위 내에 들어가도록 상기 변환 스테이지 출력을 클리핑함으로써 상기 변환 스테이지 출력을 제한하게 하는 코드를 더 포함하는, 비일시적 컴퓨터-판독가능 매체.

청구항 20

제 17 항에 있어서,

실행될 경우, 상기 장치로 하여금, 상기 제한된 변환 스테이지 출력과 상기 변환 스테이지 출력 사이의 편차를 결정하게 하는 코드를 더 포함하는, 비일시적 컴퓨터-판독가능 매체.

청구항 21

제 20 항에 있어서,

상기 편차는 상기 제한된 변환 스테이지 출력과 상기 변환 스테이지 출력 사이의 차이를 포함하는, 비일시적 컴퓨터-판독가능 매체.

청구항 22

제 20 항에 있어서,

실행될 경우, 상기 장치로 하여금, 상기 편차가 미리 결정된 임계값보다 클 때 계수들의 서브세트를 적어도 하나의 변환 스테이지 내에서 재계산하게 하는 코드를 더 포함하는, 비일시적 컴퓨터-판독가능 매체.

청구항 23

제 22 항에 있어서,

실행될 경우, 상기 장치로 하여금, 적어도 재-계산된 상기 계수들의 서브세트를 이용하여 상기 비디오 정보를 변환하게 하는 코드를 더 포함하는, 비일시적 컴퓨터-판독가능 매체.

청구항 24

비디오를 인코딩하는 장치로서,

비디오 정보를 저장하는 수단;

변환을 다수의 변환 스테이지들로 분해하는 수단;

각각의 변환 스테이지에서의 변환 스테이지 출력을 결정하기 위해 상기 다수의 스테이지들을 이용하여 상기 비디오 정보를 변환하는 수단;

상기 각각의 변환 스테이지에서의 상기 변환 스테이지 출력을 미리 결정된 비트 심도로 제한하는 수단; 및

상기 다수의 스테이지들의 최종 스테이지의 제한된 변환 출력에 대한 동작들을 수행하는 수단을 포함하며,

상기 동작들은 단지 상기 미리 결정된 비트 심도를 가지는 데이터와 함께 사용하기 위해 이용가능한, 비디오를 인코딩하는 장치.

청구항 25

제 25 항에 있어서,

메시-기반의 방법, 버터플라이 방법, 또는 짝수-홀수 분해를 이용하여 상기 변환을 분해하는 수단을 더 포함하는, 비디오를 인코딩하는 장치.

청구항 26

제 25 항에 있어서,

클리핑된 상기 변환 스테이지 출력이 반드시 미리 결정된 범위 내에 들어가도록 상기 변환 스테이지 출력을 클리핑함으로써 상기 변환 스테이지 출력을 제한하는 수단을 더 포함하는, 비디오를 인코딩하는 장치.

청구항 27

제 25 항에 있어서,

상기 제한된 변환 스테이지 출력과 상기 변환 스테이지 출력 사이의 편차를 결정하는 수단을 더 포함하는, 비디오를 인코딩하는 장치.

청구항 28

제 27 항에 있어서,

상기 편차는 상기 제한된 변환 스테이지 출력과 상기 변환 스테이지 출력 사이의 차이를 포함하는, 비디오를 인코딩하는 장치.

청구항 29

제 27 항에 있어서,

상기 편차가 미리 결정된 임계값보다 클 때 계수들의 서브세트를 적어도 하나의 변환 스테이지 내에서 재계산하

는 수단을 더 포함하는, 비디오를 인코딩하는 장치.

청구항 30

제 30 항에 있어서,

적어도 재-계산된 상기 계수들의 서브세트를 이용하여 상기 비디오 정보를 변환하는 수단을 더 포함하는, 비디오를 인코딩하는 장치.

발명의 설명

기 술 분 야

[0001] 본 개시물은 비디오 인코딩에 관한 것이다.

배 경 기 술

[0002] 디지털 비디오 능력들은, 디지털 텔레비전들, 디지털 직접 브로드캐스트 시스템들, 무선 브로드캐스트 시스템들, 개인 휴대정보 단말기들 (PDA들), 랩탑 또는 데스크탑 컴퓨터들, 태블릿 컴퓨터들, e-북 리더들, 디지털 카메라들, 디지털 리코딩 디바이스들, 디지털 미디어 플레이어들, 비디오 게이밍 디바이스들, 비디오 게임 콘솔들, 셀룰러 또는 위성 무선 전화기들, 스마트폰들, 원격 화상회의 디바이스들, 비디오 스트리밍 디바이스들 등을 포함한, 광범위한 디바이스들에 포함될 수 있다. 디지털 비디오 디바이스들은 MPEG-2, MPEG-4, ITU-T H.263, ITU-T H.264/MPEG-4, 파트 10, AVC (Advanced Video Coding) 에 의해 정의된 표준들, HEVC (High Efficiency Video Coding), 및 이런 표준들의 확장판들에서 설명되는 것들과 같은, 비디오 코딩 기법들을 구현한다. 비디오 디바이스들은 이러한 비디오 코딩 기법들을 구현함으로써, 디지털 비디오 정보를 좀더 효율적으로 송신하거나, 수신하거나, 인코딩하거나, 디코딩하거나, 및/또는 저장할 수도 있다.

[0003] 비디오 코딩 기법들은 비디오 시퀀스들에 고유한 리던던시를 감소시키거나 또는 제거하기 위해, 공간 (인트라-픽처) 예측 및/또는 시간 (인터-픽처) 예측을 포함한다. 블록-기반 비디오 코딩에 있어, 비디오 슬라이스 (예컨대, 비디오 프레임 또는 비디오 프레임의 부분) 은 비디오 블록들로 파티셔닝될 수도 있으며, 이 비디오 블록들은 또한 트리블록들, 코딩 유닛들 (CU들), 및/또는 코딩 노드들로서 지칭될 수도 있다. CU들은 CU 에 대한 예측 비디오 데이터를 결정하기 위해 하나 이상의 예측 유닛들 (PU들) 로 더 파티셔닝될 수도 있다. 비디오 압축 기법들은 또한 CU들을, 코딩될 비디오 블록과 예측 비디오 데이터 사이의 차이를 나타내는 잔차 비디오 블록 데이터의 하나 이상의 변환 유닛들 (TU들) 로 파티셔닝할 수도 있다 2차원 이산 코사인 변환 (DCT) 과 같은, 선형 변환들이, 잔차 비디오 블록 데이터를 픽셀 도메인으로부터 주파수 도메인으로 변환하여 추가적인 압축을 달성하기 위해, TU 에 적용될 수도 있다. 또한, 픽처의 인트라-코딩된 (I) 슬라이스에서의 비디오 블록들은 동일한 픽처의 이웃하는 블록들에서의 참조 샘플들에 대한 공간 예측을 이용하여 인코딩될 수도 있다. 픽처의 인터-코딩된 (P 또는 B) 슬라이스에서의 비디오 블록들은 동일한 픽처의 이웃하는 블록들에서의 참조 샘플들에 대한 공간 예측, 또는 다른 참조 픽처들에서의 참조 샘플들에 대한 시간 예측을 이용할 수도 있다. 픽처들은 프레임들로 지칭될 수 있으며, 참조 픽처들은 참조 프레임들로서 지칭될 수도 있다.

[0004] 공간 또는 시간 예측은 코딩될 블록에 대한 예측 블록을 초래한다. 잔차 데이터는 코딩될 원래 블록과 예측 블록 사이의 픽셀 차이들을 나타낸다. 인터-코딩된 블록은 예측 블록을 형성하는 참조 샘플들의 블록을 가리키는 모션 벡터, 및 코딩된 블록과 예측 블록 사이의 차이를 나타내는 잔차 데이터에 따라서 인코딩된다. 인트라-코딩된 블록은 인트라-코딩 모드 및 잔차 데이터에 따라서 인코딩된다. 추가적인 압축을 위해, 잔차 데이터는 픽셀 도메인으로부터 변환 도메인으로 변환되어, 잔차 변환 계수들을 초래할 수도 있으며, 이 잔차 변환 계수는 그후 양자화될 수도 있다. 처음에 2차원 어레이로 배열된, 양자화된 변환 계수들은 변환 계수들의 1차원 벡터를 발생하기 위해 스캐닝될 수도 있으며, 엔트로피 인코딩이 더욱 더 많은 압축을 달성하기 위해 적용될 수도 있다.

[0005] AVC 와 같은 구 비디오 표준들에서, 순방향 변환 및 역변환 사이즈 (예컨대, 4x4 및 8x8) 는 비디오 인코딩 성능에 있어 병목으로서 작용하지 않았다. 그러나, 좀더 현대의 HEVC 표준은 최고 16x16 및 32x32 순방향 변환 및 역변환 사이즈들을 이용하는데, 이것은 HEVC 프로세스에 있어 제한 인자로서 작용한다. 더 큰 변환들은 픽셀 도메인으로부터 계수 도메인으로 변환할 때 프로세싱하는데 더 많은 복잡성 및 사이클들을 필요로 한다. 코딩 효율을 도모하기 위해, 표준은 비디오 인코더에서 큰 순방향 변환 벡터들을 다수의 스테이지들로 분해하여 (예컨대, "메시-기반의 방법", "버터플라이 방법" 또는 "짝수-홀수 분해") 각각의 스테이지에서 내

부 비트 심도를 제한하는 프로세스로부터 이점을 취할 것이다. 본원에서 개시된 기법들의 일부 이점들은 비디오 인코더에서 큰 순방향 변환 벡터들을 다수의 스테이지들로 분해하고 각각의 스테이지에서 내부 비트 심도를 제한함으로써 비디오 인코딩 동안 코딩 효율을 향상시키고 계산 리소스 요구사항들을 감소시키는 것에 관한 것이다.

발명의 내용

과제의 해결 수단

[0006] 일반적으로, 본 개시물은 큰 순방향 변환들을 다수의 스테이지들로 분해하고 (예컨대, 순방향 변환을 구현하는 메시-기반의 방법) 각각의 스테이지에서의 내부 비트 심도를 계산 효율적인 명령 세트들을 수용할 수도 있는 레벨로 제한함으로써, 비디오 인코딩 성능을 향상시키는 것과 관련된 기법들을 설명한다. 예를 들어, 변환을 위한 입력 비트 심도가 9-비트일 수도 있고 시작 내부 비트 심도가 16-비트보다 클 수도 있지만, 비디오 인코더는 포화 로직 (saturation logic) 을 이용하여 변환의 내부 비트 심도를 16-비트로 제한하도록 (예컨대, 내부 값들을 16-비트로 클리핑하도록) 구성될 수도 있다. 비디오 품질의 손실을 방지하기 위해, 비디오 인코더는 편차들 (예컨대, 에러들) 의 레벨을 측정하고, 레벨을 임계치와 비교하고, 그리고 편차들의 레벨이 임계치를 초과하면 변환된 계수들의 서브세트를 재계산하도록 더 구성될 수도 있다.

[0007] 하나 이상의 예들의 세부 사항들이 첨부도면 및 아래의 상세한 설명에서 개시된다. 다른 특성들, 목적들, 및 이점들은 설명 및 도면들로부터, 그리고 청구항들로부터 명백히 알 수 있을 것이다.

[0008] 본 개시물에서 설명되는 기술요지의 일 양태는 비디오 정보를 저장하도록 구성된 메모리를 포함하는 비디오 인코더를 제공한다. 비디오 인코더는 메모리와 통신하는 프로세서를 더 포함한다. 프로세서는 변환을 다수의 변환 스테이지들로 분해하도록 구성된다. 프로세서는 각각의 변환 스테이지에서의 변환 스테이지 출력을 결정하기 위해 다수의 스테이지들을 이용하여 비디오 정보를 변환하도록 더 구성된다. 프로세서는 각각의 변환 스테이지에서의 변환 스테이지 출력을 미리 결정된 비트 심도로 제한하도록 더 구성된다. 프로세서는 다수의 스테이지들의 최종 스테이지의 제한된 변환 출력에 대한 동작들을 수행하도록 더 구성되며, 동작들은 단지 미리 결정된 비트 심도를 가지는 데이터와 함께 사용하는데 이용가능하다.

[0009] 본 개시물에서 설명되는 기술요지의 다른 양태는 비디오를 인코딩하는 방법을 제공한다. 본 방법은 비디오 정보를 저장하는 단계를 포함한다. 본 방법은 변환을 다수의 변환 스테이지들로 분해하는 단계를 더 포함한다. 본 방법은 각각의 변환 스테이지에서의 변환 스테이지 출력을 결정하기 위해 다수의 스테이지들을 이용하여 비디오 정보를 변환하는 단계를 더 포함한다. 본 방법은 각각의 변환 스테이지에서의 변환 스테이지 출력을 미리 결정된 비트 심도로 제한하는 단계를 더 포함한다. 본 방법은 다수의 스테이지들의 최종 스테이지의 제한된 변환 출력에 대한 동작들을 수행하는 단계를 더 포함하며, 동작들은 단지 미리 결정된 비트 심도를 가지는 데이터와 함께 사용하는데 이용가능하다.

[0010] 본 개시물에서 설명되는 기술요지의 다른 양태는 비일시적 컴퓨터-판독가능 매체를 제공한다. 매체는 실행될 때, 장치로 하여금, 비디오 정보를 저장하도록 하는 코드를 포함한다. 매체는 실행될 때, 장치로 하여금, 변환을 다수의 변환 스테이지들로 분해하도록 하는 코드를 더 포함한다. 매체는 실행될 때, 장치로 하여금, 각각의 변환 스테이지에서의 변환 스테이지 출력을 결정하기 위해 다수의 스테이지들을 이용하여 비디오 정보를 변환하도록 하는 코드를 더 포함한다. 매체는 실행될 때, 장치로 하여금, 각각의 변환 스테이지에서의 변환 스테이지 출력을 미리 결정된 비트 심도로 제한하도록 하는 코드를 더 포함한다. 매체는 실행될 때, 장치로 하여금, 다수의 스테이지들의 최종 스테이지의 제한된 변환 출력에 대한 동작들을 수행하도록 하는 코드를 더 포함하며, 동작들은 단지 미리 결정된 비트 심도를 가지는 데이터와 함께 사용하는데 이용가능하다.

[0011] 본 개시물에서 설명되는 기술요지의 다른 양태는 비디오를 인코딩하는 장치를 제공한다. 장치는 비디오 정보를 저장하는 수단을 포함한다. 이 장치는 변환을 다수의 변환 스테이지들로 분해하는 수단을 더 포함한다. 이 장치는 각각의 변환 스테이지에서의 변환 스테이지 출력을 결정하기 위해 다수의 스테이지들을 이용하여 비디오 정보를 변환하는 수단을 더 포함한다. 이 장치는 각각의 변환 스테이지에서의 변환 스테이지 출력을 미리 결정된 비트 심도로 제한하는 수단을 더 포함한다. 이 장치는 다수의 스테이지들의 최종 스테이지의 제한된 변환 출력에 대한 동작들을 수행하는 수단을 더 포함하며, 동작들은 단지 미리 결정된 비트 심도를 가지는 데이터와 함께 사용하는데 이용가능하다.

도면의 간단한 설명

[0012] 도 1 은 본 개시물에서 설명하는 양태들에 따른 기법들을 이용할 수도 있는 예시적인 비디오 인코딩 및 디코딩 시스템을 예시하는 블록도이다.

도 2 는 본 개시물에서 설명하는 양태들에 따른 기법들을 구현할 수도 있는 예시적인 비디오 인코더의 예를 예시하는 블록도이다.

도 3 은 본 개시물에서 설명하는 양태들에 따른 기법들을 구현할 수도 있는 예시적인 비디오 디코더의 예를 예시하는 블록도이다.

도 4 는 예를 들어, 메시-기반의 계산들을 이용하여 저 복잡성 순방향 변환의 방법의 플로우차트를 예시한다.

도 5 는 저 복잡성 순방향 변환을 위한 방법의 플로우차트를 예시한다.

도면들에 예시된 여러 특징부들은 실제 척도로 도시되지 않을 수도 있다. 따라서, 여러 특징부들의 치수들이 명료성을 위해 임의적으로 확장되거나 또는 감소될 수도 있다. 게다가, 도면들의 일부는 주어진 시스템, 방법 또는 디바이스의 구성요소들 모두를 도시하지는 않을 수도 있다. 마지막으로, 유사한 도면부호들이 명세서 및 도면들 전반에 걸쳐서 유사한 특징부들을 표시하기 위해 사용될 수도 있다.

발명을 실시하기 위한 구체적인 내용

[0013] 본 개시물에서 설명하는 기법들은 일반적으로 특히 고-효율 비디오 코딩 (HEVC) 표준 및 그의 확장판들에 대해, 비디오 인코딩 동안 순방향 변환들에 관한 것이다.

[0014] 비디오 코딩 표준들은 ITU-T H.261, ISO/IEC MPEG-1 Visual, ITU-T H.262 또는 ISO/IEC MPEG-2 Visual, ITU-T H.263, ISO/IEC MPEG-4 Visual 및 그의 스케일러블 비디오 코딩 (SVC) 및 멀티뷰 비디오 코딩 (MVC) 확장판들을 포함한, (또한, ISO/IEC MPEG-4 AVC 로서 알려진) ITU-T H.264 를 포함한다. 게다가, ITU-T 비디오 코딩 전문가 그룹 (VCEG) 와 ISO/IEC 동화상 전문가 그룹 (MPEG) 의 비디오 코딩에 관한 합동 연구팀 (JCT-VC) 에 의해 개발된, 새로운 비디오 코딩 표준, 즉 고효율 비디오 코딩 (HEVC) 이 있다.

[0015] 위에서 언급한 바와 같이, HEVC 표준은 (예컨대, HEVC 인코더에서) 최고 32x32 사이즈의 순방향 변환들 및 역변환들을 이용하며, 반면 AVC 표준은 단지 최고 8x8 변환 사이즈를 이용한다. 더 큰 변환 사이즈들은 HEVC 에서 큰 코드 블록들의 코딩 효율을 증가시키며; 그러나, 그들은 또한 더 작은 변환 사이즈들을 이용하는 것에 비해, 복잡성, 컴퓨팅 사이클들, 및 프로세싱 시간을 증가시킨다. 본 개시물에서 설명되는 방법들은 인코더가 비디오 정보를 픽셀 도메인으로부터 계수 도메인으로 변환할 때 요구되는 증가된 복잡성 및 사이클들을 감소시킬 수도 있다. 예를 들어, 어떤 방법들은 비디오 인코더에서 큰 순방향 변환 벡터들을 다수의 스테이지들로 분해하고 (예컨대, 메시-기반의 방법) 각각의 스테이지에서의 내부 비트 심도를 제한하는 것을 포함한다.

[0016] 일부 구현예들에서, 잔차 도메인으로부터 계수 도메인으로 변환하려는 목적들을 위해 (예컨대, 메시-기반, 또는 "버터플라이", 즉, 순방향 변환을 구현하는 방법을 이용하여) 큰 순방향 변환들을 다수의 스테이지들로 분해하는 것은 매트릭스 곱셈 방법을 이용하는 것보다 좀더 효율적인 프로세싱을 초래할 수도 있다. 메시-기반의 NxN 변환 구현예의 일 예는 아래 부록 A 의 코드에 예시된다. 일 구현예에서, 변환 유닛 (예컨대, 16x16 변환) 은 루마 값을 각각 나타낼 수도 있는 256개의 잔차 소스 픽셀들 (예컨대, 부록 A 에서 pSrc) 에서 시작할 수도 있다. 프로세서 또는 인코더 (예컨대, 도 2 의 인코더 (20) 의 변환 프로세싱 유닛 (52)) 는 그후 각각의 소스 픽셀에 대해 하나씩, 256개의 출력 계수들 (예컨대, 부록 A 에서 pDst) 을 결정할 수도 있다. 제 1 스테이지 이후, 프로세서 또는 인코더 (예컨대, 변환 프로세싱 유닛 (52)) 는 그후 2개의 픽셀들의 128개의 합계들 및 2개의 픽셀들의 128개의 차이들 (예컨대, 부록 A 의 nE들 및 nO들) 을 결정할 수도 있다. 제 2 프로세싱 스테이지 동안, 프로세서 또는 인코더는 nE 및 nO 합계들 및 차이들을 이용하여 쌍의 합계들 및 차이들의 합계를 결정할 수도 있다. 4개의 스테이지들 이후, 프로세서 또는 인코더는 출력 계수 (예컨대, pDst) 를 발생시킬 수도 있다. (부록 A 에 예시된 바와 같이) 이 메시-기반의 방법을 이용하여, 풀-사이즈 변환은 여러 더 작은, 덜 복잡한 변환들로 분해될 수도 있으며, 이것은 함께 곱해질 경우 풀-사이즈 변환을 다시 발생시킬 것이다.

[0017] 메시 방법, 예컨대 위에서 설명된 메시 방법 또는 임의의 다른 분해 방법을 수행한 후, 본 개시물에서 설명되는 방법들은, 각각의 스테이지에서 내부 비트 심도를 어떤 레벨로 제한하여 프로세서 또는 인코더로 하여금 그 레벨에 대해 추가적인 계산 효율적인 명령 세트들을 이용가능하게 하도록 프로세서 또는 인코더를 구성할 수도 있

다. 실제로, 어떤 인코더들 및 디코더들은 어떤 비트 심도들을 가지는 입력들과 함께 사용하는데 단지 이용 가능한 계산 효율적인 명령 세트들을 포함한다. 예를 들어, 변환을 위한 입력 비트 심도가 9-비트일 수도 있고 시작 내부 비트 심도가 16-비트보다 더 클 수도 있지만, 비디오 인코더는 변환의 각각의 분해된 스테이지의 내부 비트 심도 (예컨대, 단지 변환 출력 비트 심도보다는 변환 동작 동안의 비트 심도) 를 16-비트로 제한하도록 구성될 수도 있다. 각각의 스테이지의 내부 비트 심도를 16-비트로 제한한 상태에서, 프로세서는 16-비트 동작들 (예컨대, ARM 아키텍처, 진보된 SIMD (NEON), 디지털 신호 프로세싱 (DSP), 등) 과 함께 사용하기 위해 특별히 설계된 계산 효율적인 명령 세트들을 이용할 수도 있다. 일 구현예에서, 비디오 인코더는 포화 로직을 이용하여 각각의 스테이지에서 내부 비트 심도를 제한할 (예컨대, 내부 값들을 16-비트로 클리핑할) 수도 있다.

[0018] 일부의 경우, 변환 스테이지들에서 비트 심도를 제한하는 것은 최종 비디오 품질의 감소를 초래할 수도 있다. 이러한 결과를 방지하기 위해, 본 개시물에서 설명되는 방법들은 제한하는 프로세스에 의해 초래되는 편차들 (예컨대, 에러들) 의 레벨을 측정할 수도 있다. 방법들은 그후 편차들의 레벨을 미리 결정된 임계치 (예컨대, 내구 한계) 와 비교하고 그후 편차들의 레벨이 미리 결정된 임계치를 초과하면 변환된 계수들의 서브세트를 재계산할 수도 있다. 이 방식으로 비트 심도를 제한하고 계수들의 서브세트를 재계산함으로써, 본 개시물에서 설명되는 방법들은 비디오 인코더의 변환 프로세싱 유닛으로 하여금 더 적은 계산 리소스들을 이용하면서도 또한 비디오 품질을 유지가능하게 할 수도 있다.

[0019] 블록-기반 프로세싱 (예컨대, HEVC, 여기서, 비디오 프레임들은 비디오 블록들 또는 코딩 유닛들로 파티셔닝될 수도 있다) 을 이용하는 비디오 코덱들에서, (예컨대, 인터 또는 인트라 예측으로부터의) 예측 블록들 또는 예측 유닛들은 원래 픽셀들로부터 감산될 수도 있다. 위에서 추가로 설명된 바와 같이, 잔차 데이터는 그후 (예컨대, 추가적인 압축을 달성하기 위해) 순방향 변환들 (예컨대, 이산 코사인 변환들) 을 이용하여 잔차 변환 계수들로 변환되고, 양자화되고, 그리고 엔트로피 인코딩될 수도 있다. 엔트로피 인코딩은 아래에서 추가로 설명되는 여러 엔트로피 코딩 엔진들 (예컨대, CAVLC, CABAC, 등) 을 이용하여 수행될 수도 있다. 그 후에, 그리고 또한 아래에서 추가로 설명되는 바와 같이, 디코더는 그후 계수들을 엔트로피 디코딩하고, 역양자화하고, 그리고 역변환할 수도 있다. 마지막으로, 그 계수들은 복원된 픽셀들을 형성하기 위해 예측 블록들에 다시 가산될 수도 있다.

[0020] 비디오 코딩의 일 실시형태에서, 이미지 블록은 복원된 시간적으로 및/또는 공간적으로 이웃하는 블록들에서의 픽셀들을 이용하여 먼저 예측될 수도 있다. 예측 에러 (종종 "레지듀 (residue)" 로서 지칭됨) 는 그후 변환되고 양자화될 수도 있다. 예를 들어, S 가 사이즈 NxN 의 레지듀 블록이면, 변환된 블록 K 는 다음과 같이 매트릭스 곱셈을 이용하여 유도될 수도 있다:

[0021] K = A * S * B

[0022] 여기서, K, A, 및 B 는 또한 사이즈 NxN 이다. A 는 수직 변환 매트릭스이고 B 는 수평 변환 매트릭스이다. 일부 실시형태들에서, A 및 B 는 서로의 전치행렬 (transpose) 이다 (예컨대, B = A' 이며, 여기서, " ' " 는 전치행렬을 의미한다). 다른 실시형태들에서, A 및 B 는 서로의 전치행렬이 아니다. A 및 B 가 서로의 전치행렬일 때, 앞의 방정식은 다음과 같이 된다:

[0023] K = A * S * A'

[0024] 각각의 변환 (A 및 B) 은 다양한 변환들 중 임의의 변환을 포함할 수도 있다. 일부 실시형태들에서, 변환은 이산 코사인 변환 (DCT), 이산 사인 변환 (DST), Hadamard 변환, Haar 변환 등 중 하나를 포함한다.

[0025] SVC 확장판에서, 비디오 정보의 다수의 계층들이 존재할 수도 있다. 최하부 계층은 기초 계층 (BL) 으로서 기능할 수도 있으며, 최상부 계층은 향상된 계층 (EL) 또는 "향상 계층" 으로서 기능할 수도 있다. 최상부 계층과 최하부 계층 사이의 모든 계층들은 EL들 또는 BL들 중 하나 또는 그 양쪽으로서 기능할 수도 있다. SVC 는 품질 스케일러빌리티 (또는, 신호-대-잡음비, SNR), 공간 스케일러빌리티, 및/또는 시간 스케일러빌리티를 제공하는데 사용될 수도 있다. 향상된 계층은 기초 계층과는 상이한 공간 해상도를 가질 수도 있다. 현재의 블록의 예측은 SVC 에 대해 제공되는 상이한 계층들을 이용하여 수행될 수도 있다. 이러한 예측은 인터-계층 예측으로서 지칭될 수도 있다. 인터-계층 예측 방법들이 인터-계층 리던던시를 감소시키기 위해 SVC 에 이용될 수도 있다. 인터-계층 예측의 일부 예들은 인터-계층 인트라 예측, 인터-계층 모션 예측, 및 인터-계층 잔차 예측을 포함할 수도 있다. 인터-계층 인트라 예측은 기초 계층에서 병치된 블록들의 복원을 이용하여 향상 계층에서의 현재의 블록을 예측한다. 인터-계층 모션 예측은 기초 계층의 모션을 이용하여

향상 계층에서의 모션을 예측한다. 인터-계층 잔차 예측은 기초 계층의 레지듀를 이용하여 향상 계층의 레지듀를 예측한다. "인트라 BL 모드" 로서 지칭되는 향상 계층에 대한 하나의 특정의 코딩 모드는 기초 계층에서 대응하는 (예컨대, 동일한 공간 로케이션에 로케이트되는, 즉, "병치된" 으로서 종종 지칭됨) 블록들의 텍스쳐를 이용하여 예측될 수도 있는 텍스쳐를 포함한다.

[0026] 인터-계층 잔차 예측에서, 기초 계층의 레지듀는 향상 계층에서 현재의 블록을 예측하는데 사용될 수도 있다. 레지듀는 비디오 유닛과 소스 비디오 유닛에 대한 시간 예측 사이의 차이로서 정의될 수도 있다. 잔차 예측에서, 기초 계층의 레지듀는 또한 현재의 블록을 예측할 때에 고려된다. 예를 들어, 현재의 블록은 향상 계층으로부터의 레지듀, 향상 계층으로부터의 시간 예측, 및 기초 계층으로부터의 레지듀를 이용하여 복원될 수도 있다. 현재의 블록은 다음 방정식에 따라서 복원될 수도 있다:

[0027] $$\hat{Ie} = re + Pe + rb$$

[0028] 여기서, $\hat{Ie}$ 는 현재의 블록의 복원을 표시하고, re 는 향상 계층으로부터의 레지듀를 표시하고, Pe 는 향상 계층으로부터의 시간 예측을 표시하고, 그리고 rb 는 기초 계층으로부터의 레지듀 예측을 표시한다.

[0029] 차이 도메인을 이용하는 인터 코딩에 있어, 현재의 예측된 블록은, 향상 계층 참조 픽처에서의 대응하는 예측된 블록 샘플들과 스케일링된 기초 계층 참조 픽처에서의 대응하는 예측된 블록 샘플들 사이의 차이 값들에 기초하여 결정된다. 차이 값들은 차이 예측된 블록으로서 지칭될 수도 있다. 병치된 기초 계층 복원된 샘플들이 향상 계층 예측 샘플들을 획득하기 위해, 차이 예측된 블록에 추가된다.

[0030] 본 개시물에서 설명하는 기법들은 HEVC 에서의 순방향 변환들의 매트릭스 곱셈 동안 복잡한 계산 요구사항들에 관련한 이슈들을 해결할 수도 있다. 이 기법들은 인코더 및/또는 변환 프로세싱 유닛이 순방향 변환 매트릭스 곱셈을 수행할 수도 있는, 속도, 효율, 및 효능을 향상시킬 수도 있다.

[0031] 신규한 시스템들, 장치들, 및 방법들은 이하에서 첨부 도면들을 참조하여 좀더 충분히 설명된다. 본 개시물은 그러나, 많은 상이한 형태들로 구현될 수도 있으며, 본 개시물 전반에 걸쳐 제시되는 임의의 특정의 구조 또는 기능에 한정되는 것으로 해석되어서는 안된다. 대신, 이들 양태들은 본 개시물이 철저하고 완전하게 되도록, 그리고 본 개시물의 범위를 당업자들에게 충분히 전달하기 위해서 제공된다. 본원에서의 교시들에 기초하여, 당업자는 본 개시물의 범위가 본 발명의 임의의 다른 양태와 독립적으로 구현되든 그와 결합되든, 본원에서 개시된 신규한 시스템들, 장치들, 및 방법들의 임의의 양태를 포괄하도록 의도되는 것으로 이해하여야 한다. 예를 들어, 본원에서 개시된 양태들의 임의 개수의 양태를 이용하여, 장치가 구현될 수도 있거나 또는 방법이 실시될 수도 있다. 게다가, 본 발명의 범위는 본원에서 개시된 본 발명의 여러 양태들에 추가해서 또는 이 이외에, 다른 구조, 기능, 또는 구조 및 기능을 이용하여 실행되는 장치 또는 방법을 포괄하도록 의도된다. 본원에서 개시된 임의의 양태는 청구항의 하나 이상의 엘리먼트들에 의해 구현될 수도 있는 것으로 이해되어야 한다.

[0032] 특정의 양태들이 본원에서 설명되지만, 이들 양태들의 많은 변형예들 및 치환들은 본 개시물의 범위 이내 이다. 바람직한 양태들의 일부 이익들 및 이점들이 언급되지만, 본 개시물의 범위는 특유의 이점들, 용도들, 또는 목적들에 한정되는 것으로 의도되지 않는다. 대신, 본 개시물의 양태들은 상이한 무선 기술들, 시스템 구성들, 네트워크들, 및 송신 프로토콜들에 넓게 적용가능한 것으로 의도되며, 이들 중 일부가 일 예로서 도면들에 그리고 바람직한 양태들의 다음 설명에 예시된다. 상세한 설명 및 도면들은 한정하기 보다는 단지 본 개시물의 예시이며, 본 개시물의 범위는 첨부된 청구범위 및 이의 균등물들에 의해 정의된다.

[0033] 도 1 은 본 개시물에서 설명하는 양태들에 따른 기법들을 이용할 수도 있는 예시적인 비디오 인코딩 및 디코딩 시스템을 예시하는 블록도이다. 도 1 에 나타낸 바와 같이, 비디오 인코딩 및 디코딩 시스템 (10) 은 목적지 디바이스 (14) 에 의해 추후에 디코딩되는 인코딩된 비디오 데이터를 제공하는 소스 디바이스 (12) 를 포함한다. 특히, 소스 디바이스 (12) 는 비디오 데이터를 목적지 디바이스 (14) 에 컴퓨터-판독가능 매체 (16) 를 통해서 제공한다. 소스 디바이스 (12) 및 목적지 디바이스 (14) 는 데스크탑 컴퓨터들, 노트북 (예컨대, 랩탑) 컴퓨터들, 태블릿 컴퓨터들, 셋-탑 박스들, 전화기 핸드셋들 (예컨대, 스마트폰들), 텔레비전들, 카메라들, 디스플레이 디바이스들, 디지털 미디어 플레이어들, 비디오 게이밍 콘솔들, 비디오 스트리밍 디바이스들, 등을 포함한, 광범위한 디바이스들 중 임의의 디바이스를 포함할 수도 있다. 일부 경우들에서, 소스 디바이스 (12) 및 목적지 디바이스 (14) 은 무선 통신용으로 탑재될 수도 있다.

[0034] 위에서 언급한 바와 같이, 목적지 디바이스 (14) 는 디코딩되는 인코딩된 비디오 데이터를 컴퓨터-판독가능 매

체 (16) 를 통해서 수신할 수도 있다. 컴퓨터-판독가능 매체 (16) 는 인코딩된 비디오 데이터를 소스 디바이스 (12) 로부터 목적지 디바이스 (14) 로 이동시킬 수 있는 임의 종류의 매체 또는 디바이스를 포함할 수도 있다. 일 예에서, 컴퓨터-판독가능 매체 (16) 는 소스 디바이스 (12) 로 하여금 인코딩된 비디오 데이터를 직접 목적지 디바이스 (14) 로 실시간으로 송신할 수 있게 하는 통신 매체 (미도시됨) 를 포함할 수도 있다. 인코딩된 비디오 데이터는 무선 통신 프로토콜과 같은 통신 표준에 따라서 변조되어 목적지 디바이스 (14) 로 송신될 수도 있다. 통신 매체는 무선 주파수 (RF) 스펙트럼 또는 하나 이상의 물리적인 송신 라인들과 같은, 임의의 무선 또는 유선 통신 매체를 포함할 수도 있다. 통신 매체는 또한, 근거리 네트워크, 광역 네트워크, 또는 글로벌 네트워크 (예컨대, 인터넷) 와 같은 패킷-기반 네트워크의 일부를 형성할 수도 있다. 통신 매체는 라우터들, 스위치들, 기지국들, 또는 소스 디바이스 (12) 로부터 목적지 디바이스 (14) 로 통신을 용이하게 하는데 유용할 수도 있는 임의의 다른 장비를 포함할 수도 있다.

[0035] 일부 예들에서, 인코딩된 데이터는 출력 인터페이스 (22) 로부터 저장 디바이스 (미도시됨) 로 출력될 수도 있다. 이와 유사하게, 인코딩된 데이터는 입력 인터페이스 (28) 에 의해 저장 디바이스로부터 액세스될 수도 있다. 저장 디바이스는 하드 드라이브, Blu-ray 디스크들, DVD들, CD-ROM들, 플래시 메모리, 휘발성 또는 비-휘발성 메모리, 또는 인코딩된 비디오 데이터를 저장하기 위한 임의의 다른 적합한 디지털 저장 매체들과 같은 다양한 분산된 또는 로컬 액세스되는 데이터 저장 매체들 중 임의의 데이터 저장 매체를 포함할 수도 있다. 추가 예에서, 저장 디바이스는 소스 디바이스 (12) 에 의해 발생된 인코딩된 비디오를 저장할 수도 있는 파일 서버 또는 또 다른 중간 저장 디바이스에 대응할 수도 있다. 목적지 디바이스 (14) 는 저장된 비디오 데이터에 저장 디바이스로부터 스트리밍 또는 다운로드를 통해서 액세스할 수도 있다. 파일 서버는 인코딩된 비디오 데이터를 저장하고 그 인코딩된 비디오 데이터를 목적지 디바이스 (14) 로 송신하는 것이 가능한 임의 종류의 서버일 수도 있다. 예시적인 파일 서버들은 웹 서버 (예컨대, 웹사이트용), FTP 서버, NAS (network attached storage) 디바이스들, 또는 로컬 디스크 드라이브를 포함한다. 목적지 디바이스 (14) 는 인터넷 접속을 포함한, 임의의 표준 데이터 접속을 통해서, 인코딩된 비디오 데이터에 액세스할 수도 있다. 데이터 접속은 파일 서버 상에 저장되는 인코딩된 비디오 데이터에 액세스하는데 적합한, 무선 채널 (예컨대, Wi-Fi 접속), 유선 접속 (예컨대, DSL, 케이블 모뎀 등), 또는 양쪽의 조합을 포함할 수도 있다. 저장 디바이스로부터의 인코딩된 비디오 데이터의 송신은 스트리밍 송신, 다운로드 송신, 또는 이들의 조합일 수도 있다.

[0036] 본 개시물의 기법들은 무선 애플리케이션들 또는 설정들에 반드시 한정되지는 않는다. 이 기법들은 오버-디-에어 (over-the-air) 텔레비전 브로드캐스트들, 케이블 텔레비전 송신들, 위성 텔레비전 송신들, 인터넷 스트리밍 비디오 송신들, 예컨대 HTTP 를 통한 동적 적응 스트리밍 (DASH), 데이터 저장 매체 상에 인코딩된 디지털 비디오, 데이터 저장 매체 상에 저장된 디지털 비디오의 디코딩, 또는 다른 애플리케이션들과 같은, 다양한 멀티미디어 애플리케이션들 중 임의의 애플리케이션의 지원 하에 비디오 코딩에 적용될 수도 있다. 일부 예들에서, 시스템 (10) 은 비디오 스트리밍, 비디오 플레이백, 비디오 브로드캐스팅, 비디오 전화 통신, 등과 같은, 애플리케이션들을 위한 1-방향 또는 2-방향 비디오 송신을 지원하도록 구성될 수도 있다.

[0037] 도 1 의 예에서, 소스 디바이스 (12) 는 비디오 소스 (18), 비디오 인코더 (20), 및 출력 인터페이스 (22) 를 포함한다. 목적지 디바이스 (14) 는 입력 인터페이스 (28), 비디오 디코더 (30), 및 디스플레이 디바이스 (32) 를 포함한다. 본 개시물에 따르면, 소스 디바이스 (12) 의 비디오 인코더 (20) 는 다수의 표준들 또는 표준 확장판들에 부합하는 비디오 데이터를 포함하는 비트스트림을 코딩하는 기법들을 적용하도록 구성될 수도 있다. 다른 예들에서, 소스 디바이스 (12) 및 목적지 디바이스 (14) 는 다른 구성요소들 또는 배열들을 포함할 수도 있다. 예를 들어, 소스 디바이스 (12) 는 비디오 데이터를 외부 카메라와 같은 외부 비디오 소스로부터 수신할 수도 있다. 이와 유사하게, 목적지 디바이스 (14) 는 통합된 디스플레이 디바이스 (32) 보다는, 외부 디스플레이 디바이스와 인터페이스할 수도 있다.

[0038] 본 개시물의 기법들은 일반적으로 비디오 인코딩 디바이스에 의해 수행되지만, 그 기법들은 또한 "코덱" 으로서 일반적으로 지칭되는, 비디오 인코더/디코더에 의해 수행될 수도 있다. 더욱이, 본 개시물의 기법들은 또한 비디오 프리프로세서에 의해 수행될 수도 있다. 소스 디바이스 (12) 및 목적지 디바이스 (14) 는 소스 디바이스 (12) 가 목적지 디바이스 (14) 로의 송신을 위해 코딩된 비디오 데이터를 발생시키는 단지 그러한 코딩 디바이스들의 예들일 뿐이다. 일부 예들에서, 소스 디바이스 (12) 및 목적지 디바이스 (14) 는 그들 각각이 비디오 인코딩 및 디코딩 구성요소들을 포함하도록, 실질적으로 대칭적 방식으로 동작할 수도 있다. 그러므로, 시스템 (10) 은 소스 디바이스 (12) 와 목적지 디바이스 (14) 사이에, 예컨대, 비디오 스트리밍, 비디오 플레이백, 비디오 브로드캐스팅, 비디오 전화 통신, 등을 위해 1-방향 또는 2-방향 비디오 송신을 지원할 수도 있다.

[0039] 소스 디바이스 (12) 의 비디오 소스 (18) 는 비디오 카메라와 같은 비디오 캡쳐 디바이스 (미도시됨), 이전에 캡쳐된 비디오를 포함하는 비디오 아카이브, 비디오 콘텐츠 제공자로부터 비디오를 수신하기 위한 비디오 공급 인터페이스 등을 포함할 수도 있다. 추가 대안적인 예로서, 비디오 소스 (18) 는 컴퓨터 그래픽스-기반의 데이터, 또는 라이브 비디오, 아카이브된 비디오, 및 컴퓨터 발생된 비디오의 조합을 발생시킬 수도 있다. 일부의 경우, 비디오 소스 (18) 가 비디오 카메라이면, 소스 디바이스 (12) 및 목적지 디바이스 (14) 는 카메라 폰들 또는 비디오 폰들일 수도 있다. 다른 실시형태에서, 본 개시물에서 설명하는 기법들은 비디오 코딩에 일반적으로 적용가능할 수도 있으며, 무선 및/또는 유선 애플리케이션들에 적용될 수도 있다. 각 경우, 캡쳐되거나, 사전-캡쳐되거나, 또는 컴퓨터-발생된 비디오는 비디오 인코더 (20) 에 의해 인코딩될 수도 있다. 인코딩된 비디오 정보는 그후 출력 인터페이스 (22) 에 의해 컴퓨터-판독가능 매체 (16) 상으로 출력될 수도 있다.

[0040] 컴퓨터-판독가능 매체 (16) 는 무선 브로드캐스트 또는 유선 네트워크 송신과 같은 일시성 매체, 또는 하드 디스크, 플래시 드라이브, 컴팩트 디스크, 디지털 비디오 디스크, Blu-ray 디스크, 또는 다른 컴퓨터-판독가능 매체들과 같은 저장 매체들 (즉, 비일시성 저장 매체들) 을 포함할 수도 있다. 일부 예들에서, 네트워크 서버 (미도시) 는 소스 디바이스 (12) 로부터, 인코딩된 비디오 데이터를 수신하고, 인코딩된 비디오 데이터를 목적지 디바이스 (14) 로, 예컨대, 네트워크 송신, 직접 유선 통신, 등을 통해서 제공할 수도 있다. 이와 유사하게, 디스크 스탬핑 설비와 같은 매체 생산 설비의 컴퓨팅 디바이스는 인코딩된 비디오 데이터를 소스 디바이스 (12) 로부터 수신하고 그 인코딩된 비디오 데이터를 포함하는 디스크를 제조할 수도 있다. 따라서, 컴퓨터-판독가능 매체 (16) 는 여러 형태들의 하나 이상의 컴퓨터-판독가능 매체들을 포함할 수도 있다.

[0041] 목적지 디바이스 (14) 의 입력 인터페이스 (28) 는 컴퓨터-판독가능 매체 (16) 로부터 정보를 수신할 수도 있다. 컴퓨터-판독가능 매체 (16) 의 정보는 비디오 인코더 (20) 에 의해 정의된 신택스 정보를 포함할 수도 있다. 신택스 정보는 또한 비디오 디코더 (30) 에 의해 사용될 수도 있으며, 블록들 및 다른 코딩된 유닛들의 특성들 및/또는 프로세싱을 기술하는 신택스 엘리먼트들을 포함할 수도 있다. 디스플레이 디바이스 (32) 는 그 디코딩된 비디오 데이터를 사용자에게 디스플레이할 수도 있으며, 음극선관 (CRT), 액정 디스플레이 (LCD), 플라즈마 디스플레이, 유기 발광 다이오드 (OLED) 디스플레이, 또는 또 다른 유형의 디스플레이 디바이스와 같은 다양한 디스플레이 디바이스들 중 임의의 디바이스를 포함할 수도 있다.

[0042] 비디오 인코더 (20) 및 비디오 디코더 (30) 는 HEVC (High Efficiency Video Coding) 표준 또는 그의 변형예들 중 임의의 표준 (예컨대, HEVC 테스트 모델 (HM)) 과 같은, 비디오 코딩 표준에 따라서 동작할 수도 있다. 대안적으로, 비디오 인코더 (20) 및 비디오 디코더 (30) 는 ITU-T H.264 표준 (MPEG 4), 부분 10, AVC (Advanced Video Coding), ITU-T H.263, ITU-T H.262 (ISO/IEC MPEG-2 Visual), ISO/IEC MPEG-1 Visual, ITU-T H.261, 또는 임의의 이러한 표준들의 확장판들과 같은, 다른 독점 또는 산업 표준들에 따라서 동작할 수도 있다. 일부 양태들에서, 비디오 인코더 (20) 및 비디오 디코더 (30) 는 공통 데이터 스트림 또는 별개의 데이터 스트림들에서 오디오 및 비디오 양쪽의 인코딩을 처리하기 위해 오디오 인코더, 오디오 디코더, MUX-DEMUX 유닛들 (미도시됨), 또는 다른 하드웨어 및 소프트웨어와 통합될 수도 있다.

[0043] 비디오 인코더 (20) 및 비디오 디코더 (30) 각각은 하나 이상의 마이크로프로세서들, 디지털 신호 프로세서들 (DSP들), 주문형 집적회로들 (ASIC들), 필드 프로그래밍가능 게이트 어레이들 (FPGA들), 이산 로직, 소프트웨어, 하드웨어, 펌웨어 또는 임의의 이들의 조합들과 같은, 다양한 적합한 인코더 회로 중 임의의 회로로 구현될 수도 있다. 본 개시물의 기법들을 수행하는데 필요할 경우, 비디오 인코더 (20) 및/또는 비디오 디코더 (30) 는 소프트웨어에 대한 명령들을 적합한 비일시적 컴퓨터-판독가능 매체에 저장하고 그 명령들을 하드웨어에서 하나 이상의 프로세서들을 이용하여 실행할 수도 있다. 비디오 인코더 (20) 및 비디오 디코더 (30) 각각은 하나 이상의 인코더들 또는 디코더들에 포함될 수도 있으며, 이들 중 어느 쪽이든 개별 디바이스에서 결합된 인코더/디코더 (CODEC) 의 부분으로서 통합될 수도 있다. 비디오 인코더 (20) 및/또는 비디오 디코더 (30) 를 포함하는 디바이스는 집적 회로, 마이크로프로세서, 및/또는 무선 통신 디바이스, 예컨대 셀폰을 포함할 수도 있다.

[0044] HEVC 표준은 비디오 프레임 또는 픽처가 루마 샘플 및 크로마 샘플들 양쪽을 포함하는 트리블록들 또는 최대 코딩 유닛들 (LCU) 의 시퀀스로 분할될 수도 있다고 규정한다. 비트스트림 내 신택스 데이터는 LCU 에 대한 사이즈를 정의할 수도 있으며, 이 최대 코딩 유닛은 픽셀들의 개수의 관점에서 최대 코딩 유닛이다. 슬라이스는 코딩 순서에서 다수의 연속되는 트리블록들을 포함한다. 비디오 프레임 또는 픽처는 하나 이상의 슬라이스들로 파티셔닝될 수도 있다. 각각의 트리블록은 쿼드트리에 따라 코딩 유닛들 (CU들) 로 분할될 수도 있다. 일반적으로, 쿼드트리 데이터 구조는 CU 당 하나의 노드를 포함하며, 루트 노드는 트리블록에 대응한

다. CU 가 4개의 서브-CU들로 분할되면, CU 에 대응하는 노드는 4개의 리프 노드들을 포함하며, 그 리프 노드 각각은 서브-CU들 중 하나에 대응한다.

[0045] 쿼드트리 데이터 구조의 각각의 노드는 대응하는 CU 에 대한 신택스 데이터를 제공할 수도 있다. 예를 들어, 쿼드트리에서 노드는 그 노드에 대응하는 CU 가 서브-CU들로 분할되는지의 여부를 나타내는 분할 플래그를 포함할 수도 있다. CU 에 대한 신택스 엘리먼트들은 회귀적으로 정의될 수도 있으며, CU 가 서브-CU들로 분할되는지의 여부에 의존할 수도 있다. CU 가 추가로 분할되지 않으면, 리프-CU 로서 지칭된다. 본 개시물에서, 리프-CU 의 4개의 서브-CU들은 또한 원래 리프-CU 의 명시적인 분할이 없더라도 리프-CU들로 지칭될 것이다. 예를 들어, 16x16 사이즈에서 CU 가 추가로 분할되지 않으면, 4개의 8x8 서브-CU들이 또한 16x16 CU 가 전혀 분할되지 않았더라도 리프-CU들로서 지칭될 것이다.

[0046] CU 는 CU 가 사이즈 구별을 갖지 않는다는 점을 제외하고는, H.264 표준의 매크로블록과 유사한 목적을 갖는다. 예를 들어, 트리블록은 4개의 자식 노드들 (또한, 서브-CU들로서 지칭됨) 로 분할될 수도 있으며, 각각의 자식 노드는 결국 부모 노드일 수도 있으며 또 다른 4개의 자식 노드들로 분할될 수도 있다. 쿼드트리의 리프 노드로서 지칭되는, 최종, 미분할된 자식 노드는 리프-CU 로서 또한 지칭되는, 코딩 노드를 포함한다. 코딩된 비트스트림과 연관되는 신택스 데이터는 최대 CU 심도로서 지칭되는, 트리블록이 분할될 수도 있는 최대 횟수를 정의할 수도 있으며, 또한 코딩 노드들의 최소 사이즈를 정의할 수도 있다. 따라서, 비트스트림은 또한 최소 코딩 유닛 (SCU) 을 정의할 수도 있다. 본 개시물은 HEVC 의 상황에서는, CU, PU, 또는 TU, 또는 다른 표준들의 상황에서는, 유사한 데이터 구조들 (예컨대, H.264/AVC 에서의 매크로블록들 및 그의 서브-블록들) 중 임의의 것을 지칭하기 위해 용어 "블록" 을 사용한다.

[0047] CU 는 코딩 노드, 및 이 코딩 노드와 연관되는 변환 유닛들 (TU들) 및 예측 유닛들 (PU들) 을 포함한다. CU 의 사이즈는 코딩 노드의 사이즈에 대응하며 형상이 정사각형이어야 한다. CU 의 사이즈는 8x8 픽셀들로부터 64x64 픽셀들의 최대치, 또는 일부의 경우, 그 초과의 픽셀을 가지는 트리블록의 사이즈까지 이를 수도 있다. 각각의 CU 는 하나 이상의 PU들 및 하나 이상의 TU들을 포함할 수도 있다. CU 와 연관되는 신택스 데이터는 예를 들어, 하나 이상의 PU들로의 CU 의 파티셔닝을 기술할 수도 있다. 파티셔닝 모드들은 CU 가 스킵되는지 또는 직접 모드 인코딩될지, 인트라-예측 모드 인코딩될지, 또는 인터-예측 모드 인코딩될지 여부의 사이에서 상이할 수도 있다. PU들은 형상이 비-정사각형으로 파티셔닝될 수도 있다. CU 와 연관되는 신택스 데이터는 또한 예를 들어, 쿼드트리에 따른 하나 이상의 TU들로의 CU 의 파티셔닝을 기술할 수도 있다. TU 는 형상이 정사각형 또는 비-정사각형 (예컨대, 직사각형) 일 수 있다.

[0048] HEVC 표준은 TU들에 따라서 변환들을 허용하며, 이 TU들은 상이한 CU들에 대해 상이할 수도 있다. TU들은 일반적으로 파티셔닝된 LCU 에 대해 정의된 주어진 CU 내 PU들의 사이즈에 기초하여 사이징되지만, 이것이 항상 그런 것은 아니다. TU들은 일반적으로 PU들과 동일한 사이즈이거나 또는 그보다 작다. 일부 예들에서, CU 에 대응하는 잔차 샘플들은 "잔차 쿼드 트리" (RQT) 로서 알려진 쿼드트리 구조를 이용하여 더 작은 유닛들로 세분될 수도 있다. RQT 의 리프 노드들은 변환 유닛들 (TU들) 로서 지칭될 수도 있다. TU들과 연관되는 픽셀 차이 값들은 변환 계수들을 발생하기 위해 변환될 수도 있으며, 그 변환 계수들은 양자화될 수도 있다.

[0049] 리프-CU 는 하나 이상의 예측 유닛들 (PU들) 을 포함할 수도 있다. 일반적으로, PU 는 대응하는 CU 의 모두 또는 부분에 대응하는 공간 영역을 나타내며, PU 에 대한 참조 샘플을 취출하기 위한 데이터를 포함할 수도 있다. 더욱이, PU 는 예측에 관련된 데이터를 포함한다. 예를 들어, PU 가 인트라-모드 인코딩될 때, PU 에 대한 데이터는 잔차 쿼드트리 (RQT) 에 포함될 수도 있으며, PU 에 대응하는 TU 에 대한 인트라-예측 모드를 기술하는 데이터를 포함할 수도 있다. 또 다른 예로서, PU 가 인터-모드 인코딩될 때, PU 는 PU 에 대한 하나 이상의 모션 벡터들을 정의하는 데이터를 포함할 수도 있다. PU 에 대한 모션 벡터를 정의하는 데이터는 예를 들어, 모션 벡터의 수평 성분, 모션 벡터의 수직 성분, 모션 벡터에 대한 해상도 (예컨대, 1/4 픽셀 정밀도 또는 1/8 픽셀 정밀도), 모션 벡터가 가리키는 참조 픽처, 및/또는 모션 벡터에 대한 참조 픽처 리스트 (예컨대, List 0, List 1, 또는 List C) 를 기술할 수도 있다.

[0050] 하나 이상의 PU들을 갖는 리프-CU 는 또한 하나 이상의 변환 유닛들 (TU들) 을 포함할 수도 있다. 변환 유닛들은 위에서 설명한 바와 같이, RQT (또한, TU 쿼드트리 구조로서 지칭됨) 를 이용하여 규정될 수도 있다. 예를 들어, 분할 플래그는 리프-CU 가 4개의 변환 유닛들로 분할되는지 여부를 나타낼 수도 있다. 그 후, 각각의 변환 유닛은 추가적인 서브-TU들로 추가로 분할될 수도 있다. TU 가 추가로 분할되지 않을 때, 리프-CU 로서 지칭될 수도 있다. 일반적으로, 인트라 코딩에 있어, 리프-CU 에 속하는 모든 리프-TU들은 동일한

인트라 예측 모드를 공유한다. 즉, 동일한 인트라-예측 모드가 일반적으로 리프-CU 의 모든 TU들에 대해 예측된 값들을 계산하기 위해 적용된다. 인트라 코딩에 있어, 비디오 인코더는 각각의 리프-TU 에 대한 잔차 값을 인트라 예측 모드를 이용하여, TU 에 대응하는 CU 의 부분과 원래 블록 사이의 차이로서 계산할 수도 있다. TU 는 PU 의 사이즈로 반드시 제한될 필요는 없다. 따라서, TU들은 PU 보다 더 크거나 또는 더 작을 수도 있다. 인트라 코딩에 있어, PU 는 동일한 CU 에 대한 대응하는 리프-TU 와 병치될 수도 있다. 일부 예들에서, 리프-TU 의 최대 사이즈는 대응하는 리프-CU 의 사이즈에 대응할 수도 있다.

[0051] 더욱이, 리프-CU들의 TU들은 또한 잔차 쿼드트리들 (RQT들) 로서 지칭되는, 개별 쿼드트리 데이터 구조들과 연관될 수도 있다. 즉, 리프-CU 는 리프-CU 가 어떻게 TU들로 파티셔닝되는 지를 나타내는 쿼드트리를 포함할 수도 있다. TU 쿼드트리의 루트 노드는 일반적으로 리프-CU 에 대응하는 반면, CU 쿼드트리의 루트 노드는 일반적으로 트리블록 (또는, LCU) 에 대응한다. 분할되지 않은 RQT 의 TU들은 리프-TU들로서 지칭된다. 일반적으로, 본 개시물은 달리 언급하지 않는 한, 리프-CU 및 리프-TU 를 지칭하기 위해, 각각 용어들 CU 및 TU 를 사용한다.

[0052] 비디오 시퀀스는 일반적으로 비디오 프레임들 또는 픽처들의 시리즈를 포함한다. 픽처들의 그룹 (GOP) 은 일반적으로 비디오 픽처들의 하나 이상의 시리즈를 포함한다. GOP 는 GOP 의 헤더에, 픽처들의 하나 이상의 헤더에, 또는 다른 곳에, GOP 에 포함된 다수의 픽처들을 기술하는 신택스 데이터를 포함할 수도 있다. 픽처의 각각의 슬라이스는 개별 슬라이스에 대한 인코딩 모드를 기술하는 슬라이스 신택스 데이터를 포함할 수도 있다. 도 1 의 비디오 인코더 (20) 는 비디오 데이터를 인코딩하기 위해 개개의 비디오 슬라이스들 내 비디오 블록들에 대해 동작할 수도 있다. 비디오 블록은 CU 내 코딩 노드에 대응할 수도 있다. 비디오 블록들은 고정 또는 가변 사이즈들을 가질 수도 있으며, 규정된 코딩 표준에 따라서 사이즈가 상이할 수도 있다.

[0053] HEVC 는 여러 PU 사이즈들에서 예측을 지원한다. 특정의 CU 의 사이즈가 2Nx2N 이라고 가정하면, HEVC 는 2Nx2N 또는 NxN 의 PU 사이즈들에서는 인트라-예측을, 그리고 2Nx2N, 2NxN, Nx2N, 또는 NxN 의 대칭적인 PU 사이즈들에서는 인터-예측을 지원한다. HEVC 는 또한 2NxnU, 2NxnD, nLx2N, 및 nRx2N 의 PU 사이즈들에서 인터-예측에 대해 비대칭적인 파티셔닝을 지원한다. 비대칭적인 파티셔닝에서, CU 의 하나의 방향은 파티셔닝되지 않지만, 다른 방향은 25% 및 75% 로 파티셔닝된다. 25% 파티션에 대응하는 CU 의 부분은 "상부 (Up)", "하부 (Down)", "좌측 (Left)", 또는 "우측 (Right)" 의 표시가 뒤따르는 "n" 으로 표시된다. 따라서, 예를 들어, "2NxnU" 는 상부에서 2Nx0.5N PU 로 그리고 하부에서 2Nx1.5N PU 로 수평으로 파티셔닝된 2Nx2N CU 를 지칭한다.

[0054] 본 개시물에서, "NxN" 및 "N 바이 N" 은 수직 및 수평 치수들의 관점에서 비디오 블록의 픽셀 치수들, 예컨대, 16x16 픽셀들 또는 16 바이 16 픽셀들을 지칭하기 위해 상호교환가능하게 사용될 수도 있다. 일반적으로, 16x16 블록은 수직 방향으로 16개의 픽셀들 (y = 16) 및 수평 방향으로 16개의 픽셀들 (x = 16) 을 가질 것이다. 이와 유사하게, NxN 블록은 수직 방향으로 N 개의 픽셀들 및 수평 방향으로 N 개의 픽셀들을 가질 수도 있으며, 여기서 N 은 음이 아닌 정수 값을 나타낸다. 블록에서 픽셀들은 로우들 및 칼럼들로 배열될 수도 있다. 더욱이, 블록들은 수직 방향에서와 수평 방향에서 동일한 수의 픽셀들을 반드시 갖지 않을 수도 있다. 예를 들어, 블록들은 NxM 픽셀들을 포함할 수도 있으며, 여기서 M 은 반드시 N 과 같을 필요는 없다.

[0055] CU 의 PU들을 이용한 인트라-예측 또는 인터-예측 코딩 이후, 비디오 인코더 (20) 는 CU 의 TU들에 대한 잔차 데이터를 계산할 수도 있다. PU들은 공간 도메인 (또한, 픽셀 도메인으로 지칭됨) 에서 예측 픽셀 데이터를 발생하는 방법 또는 모드를 기술하는 신택스 데이터를 포함할 수도 있으며, TU들은, 예를 들어, 이산 코사인 변환 (DCT), 정수 변환, 웨이블릿 변환, 또는 개념적으로 유사한 변환과 같은 변환의 잔차 비디오 데이터로의 적용 이후 변환 도메인에서의 계수들을 포함할 수도 있다. 잔차 데이터는 미인코딩된 픽처의 픽셀들과 PU들에 대응하는 예측 값들 사이의 픽셀 차이들에 대응할 수도 있다. 비디오 인코더 (20) 는 CU 에 대한 잔차 데이터를 포함하는 TU들을 형성하고, 그후 그 TU들을 변환하여, 그 CU 에 대한 변환 계수들을 발생할 수도 있다.

[0056] 변환 계수들을 생성하는 임의의 변환들 이후, 비디오 인코더 (20) 는 변환 계수들의 양자화를 수행할 수도 있다. 양자화는 그의 최광의의 일반적인 의미를 가지도록 의도된 광의의 용어이다. 일 실시형태에서, 양자화는 계수들을 나타내는데 사용되는 데이터의 양을 가능한 한 감소시켜 추가적인 압축을 제공하기 위해 변환 계수들이 양자화되는 프로세스를 지칭한다. 양자화 프로세스는 그 계수들의 일부 또는 모두와 연관되는 비트 심도를 감소시킬 수도 있다. 예를 들어, n-비트 값은 양자화 동안 m-비트 값까지 절사될 수도 있으며, 여기서, n 은 m 보다 더 크다.

[0057] 양자화 이후, 비디오 인코더는 변환 계수들을 스캐닝하여, 양자화된 변환 계수들을 포함하는 2차원 매트릭스로

부터 1차원 벡터를 발생할 수도 있다. 스캐닝은 어레이의 앞부분에 더 높은 에너지 (따라서, 더 낮은 주파수) 계수들을 배치하고, 그리고 어레이의 뒷부분에 더 낮은 에너지 (따라서, 더 높은 주파수) 계수들을 배치하도록 설계될 수도 있다. 일부 예들에서, 비디오 인코더 (20) 는 엔트로피 인코딩될 수 있는 직렬화된 벡터를 발생하기 위해, 미리 정의된 스캐닝 순서를 이용하여, 양자화된 변환 계수들을 스캐닝할 수도 있다. 다른 예들에서, 비디오 인코더 (20) 는 적응적 스캐닝을 수행할 수도 있다. 양자화된 변환 계수들을 스캐닝하여 1차원 벡터를 형성한 후, 비디오 인코더 (20) 는 예컨대, 컨텍스트-적응 가변 길이 코딩 (CAVLC), 컨텍스트-적응 2진 산술 코딩 (CABAC), 신택스-기반 컨텍스트-적응 2진 산술 코딩 (SBAC), 확률 간격 파티셔닝 엔트로피 (PIPE) 코딩 또는 또 다른 엔트로피 인코딩 방법론에 따라서, 1차원 벡터를 엔트로피 인코딩할 수도 있다. 비디오 인코더 (20) 는 또한 비디오 데이터를 디코딩할 때에 비디오 디코더 (30) 에 의해 사용하기 위한 인코딩된 비디오 데이터와 연관되는 신택스 엘리먼트들을 엔트로피 인코딩할 수도 있다.

[0058] 비디오 인코더 (20) 는 블록-기반 신택스 데이터, 프레임-기반의 신택스 데이터, 및 GOP-기반 신택스 데이터와 같은 신택스 데이터를, 비디오 디코더 (30) 로, 예컨대, 프레임 헤더, 블록 헤더, 슬라이스 헤더, 또는 GOP 헤더로 추가로 전송할 수도 있다. GOP 신택스 데이터는 개별 GOP 에서의 다수의 프레임들을 기술할 수도 있으며, 프레임 신택스 데이터는 대응하는 프레임을 인코딩하는데 사용되는 인코딩/예측 모드를 나타낼 수도 있다.

[0059] 도 2 는 본 개시물에서 설명하는 양태들에 따른 기법들을 구현할 수도 있는 예시적인 비디오 인코더의 예를 예시하는 블록도이다. 비디오 인코더 (20) 의 유닛들 중 하나 이상이 본 개시물의 기법들 중 임의의 기법 또는 모두를 수행하도록 구성될 수도 있다. 일 예로서, 변환 프로세싱 유닛 (52) 은 본 개시물에서 설명되는 변환 기법들 중 임의의 기법 또는 모두를 수행하도록 구성될 수도 있다. 그러나, 본 개시물의 양태들은 이에 제한되지 않는다. 일부 예들에서, 본 개시물에서 설명하는 기법들은 비디오 인코더 (20) 의 여러 구성요소들 사이에 공유될 수도 있다. 일부 예들에서, 프로세서 (미도시) 는 본 개시물에서 설명되는 기법들 중 임의의 기법 또는 모두를 수행하도록 구성될 수도 있다.

[0060] 비디오 인코더 (20) 는 비디오 슬라이스들 내 비디오 블록들의 인트라-코딩 및 인터-코딩을 수행할 수도 있다. 인트라 코딩은 주어진 비디오 프레임 또는 픽처 내 비디오에서 공간 리던던시를 감소시키거나 또는 제거하기 위해, 공간 예측에 의존한다. 인터-코딩은 비디오 시퀀스의 인접 프레임들 또는 픽처들 내 비디오에서 시간 리던던시를 감소시키거나 또는 제거하기 위해, 시간 예측에 의존한다. 인트라-모드 (I 모드) 는 여러 공간 기반의 압축 모드들 중 임의의 코딩 모드를 참조할 수도 있다. 단방향 예측 (P 모드) 또는 양방향-예측 (B 모드) 과 같은 인터-모드들은 여러 시간-기반의 코딩 모드들 중 임의의 모드를 참조할 수도 있다.

[0061] 비디오 인코더 (20) 는 인코딩될 비디오 프레임 내 현재의 비디오 블록을 수신할 수도 있다. 도 2 의 예에서, 비디오 인코더 (20) 는 모드 선택 유닛 (40), 참조 프레임 메모리 (64), 합산기 (50), 변환 프로세싱 유닛 (52), 양자화 유닛 (54), 및 엔트로피 인코딩 유닛 (56) 을 포함한다. 모드 선택 유닛 (40) 은, 모션 추정 유닛 (42), 모션 보상 유닛 (44), 인트라-예측 유닛 (46), 및 파티션 유닛 (48) 을 포함한다. 비디오 블록 복원을 위해, 비디오 인코더 (20) 는 또한 역양자화 유닛 (58), 역변환 유닛 (60), 및 합산기 (62) 를 포함할 수도 있다. 디블록킹 필터 (미도시) 가 또한 블록 경계들을 필터링하여 복원된 비디오로부터 블록킹 현상 (blockiness) 아티팩트들을 제거하기 위해 포함될 수도 있다. 원할 경우, 디블록킹 필터는 일반적으로 합산기 (62) 의 출력을 필터링할 것이다. (인 루프 또는 사후 루프에서) 추가적인 필터들이 또한 디블록킹 필터에 추가하여 사용될 수도 있다. 이러한 필터들은 간결성을 위해 도시되지 않지만, 그러나 원할 경우, 합산기 (50) 의 출력을 (인-루프 필터로서) 필터링할 수도 있다.

[0062] 인코딩 프로세스 동안, 비디오 인코더 (20) 는 코딩될 비디오 프레임 또는 슬라이스를 수신할 수도 있다. 프레임 또는 슬라이스는 다수의 비디오 블록들로 분할될 수도 있다. 모션 추정 유닛 (42) 및 모션 보상 유닛 (44) 은 시간 예측을 제공하기 위해 하나 이상의 참조 프레임들에서 하나 이상의 블록들에 대해 그 수신된 비디오 블록의 인터-예측 코딩을 수행할 수도 있다. 인트라-예측 유닛 (46) 은 대안적으로, 공간 예측을 제공하기 위해, 코딩될 블록과 동일한 프레임 또는 슬라이스에서의 하나 이상의 이웃하는 블록들에 대한 수신된 비디오 블록의 인트라-예측 코딩을 수행할 수도 있다. 비디오 인코더 (20) 는 예컨대, 비디오 데이터의 각각의 블록에 대해 적합한 코딩 모드를 선택하기 위해, 다수의 코딩 패스(pass)들을 수행할 수도 있다.

[0063] 더욱이, 파티션 유닛 (48) 은 이전 코딩 패스들에서의 이전 파티셔닝 방식들의 평가에 기초하여, 비디오 데이터의 블록들을 서브-블록들로 파티셔닝할 수도 있다. 예를 들어, 파티션 유닛 (48) 은 레이트-왜곡 분석 (예컨대, 레이트-왜곡 최적화) 에 기초하여, 처음에 프레임 또는 슬라이스를 LCU들로 파티셔닝하고, LCU들의 각각을 서브-CU들로 파티셔닝할 수도 있다. 모드 선택 유닛 (40) 은 서브-CU들로의 LCU 의 파티셔닝을 나타내는

쿼드트리 데이터 구조를 추가로 발생할 수도 있다. 쿼드트리의 리프-노드 CU들은 하나 이상의 PU들 및 하나 이상의 TU들을 포함할 수도 있다.

[0064] 모드 선택 유닛 (40) 은 에러 결과들에 기초하여 코딩 모드들, 즉 인트라 또는 인터 중 하나를 선택할 수도 있으며, 그리고, 결과적인 인트라- 또는 인터-코딩된 블록을 합산기 (50) 에 제공하여 잔차 블록 데이터를 발생시킬 수도 있으며, 그리고 합산기 (62) 에 제공하여 참조 프레임으로서 사용을 위한 인코딩된 블록을 복원할 수도 있다. 모드 선택 유닛 (40) 은 또한 모션 벡터들, 인트라-모드 표시자들, 파티션 정보, 및 다른 이러한 신택스 정보와 같은 신택스 엘리먼트들을, 엔트로피 인코딩 유닛 (56) 에 제공할 수도 있다.

[0065] 모션 추정 유닛 (42) 및 모션 보상 유닛 (44) 은 고도로 통합될 수도 있지만, 개념적인 목적들을 위해 별개로 예시된다. 모션 추정 유닛 (42) 에 의해 수행되는 모션 추정은 모션 벡터들을 발생하는 프로세스이며, 이 프로세스는 비디오 블록들에 대한 모션을 추정한다. 모션 벡터는, 예를 들어, 현재의 프레임 (또는, 다른 코딩된 유닛) 내 코딩중인 현재의 블록에 대한 참조 프레임 (또는, 다른 코딩된 유닛) 내 예측 블록에 대한, 현재의 비디오 프레임 또는 픽처 내 비디오 블록의 PU 의 변위를 나타낼 수도 있다. 예측 블록은 픽셀 차이의 관점에서 코딩될 블록에 가깝게 매칭하는 것으로 발견되는 블록이며, SAD (sum of absolute difference), SSD (sum of square difference), 또는 다른 차이 메트릭들에 의해 결정될 수도 있다. 일부 예들에서, 비디오 인코더 (20) 는 참조 프레임 메모리 (64) 에 저장된 참조 픽처들의 서브-정수 픽셀 위치들에 대한 값들을 계산할 수도 있다. 예를 들어, 비디오 인코더 (20) 는 참조 픽처의 1/4 픽셀 위치들, 1/8 픽셀 위치들, 또는 다른 분수 픽셀 위치들의 값들을 내삽할 수도 있다. 따라서, 모션 추정 유닛 (42) 은 풀 픽셀 위치들 및 분수 픽셀 위치들에 대해, 모션 탐색을 수행하고, 분수 픽셀 정밀도를 가진 모션 벡터를 출력할 수도 있다.

[0066] 모션 추정 유닛 (42) 은 PU 의 위치를 참조 픽처의 예측 블록의 위치와 비교함으로써 인터-코딩된 슬라이스에서의 비디오 블록의 PU 에 대한 모션 벡터를 계산한다. 참조 픽처는 제 1 참조 픽처 리스트 (List 0) 또는 제 2 참조 픽처 리스트 (List 1) 로부터 선택될 수도 있으며, 이 리스트 각각은 참조 프레임 메모리 (64) 에 저장된 하나 이상의 참조 픽처들을 식별한다. 모션 추정 유닛 (42) 은 그 계산된 모션 벡터를 엔트로피 인코딩 유닛 (56) 및 모션 보상 유닛 (44) 으로 전송한다.

[0067] 모션 보상 유닛 (44) 에 의해 수행되는 모션 보상은 모션 추정 유닛 (42) 에 의해 결정된 모션 벡터에 기초하여 예측 블록을 페치하거나 또는 발생하는 것을 수반할 수도 있다. 또, 모션 추정 유닛 (42) 및 모션 보상 유닛 (44) 은 일부 예들에서, 기능적으로 통합될 수도 있다. 현재의 비디오 블록의 PU 에 대한 모션 벡터를 수신하자 마자, 모션 보상 유닛 (44) 은 모션 벡터가 참조 픽처 리스트들 중 하나에서 가리키는 예측 블록을 로케이트할 수도 있다. 합산기 (50) 는 이하에서 설명하는 바와 같이, 코딩중인 현재의 비디오 블록의 픽셀 값들로부터 예측 블록의 픽셀 값들을 감산하여 픽셀 차이 값들을 형성함으로써, 잔차 비디오 블록을 형성할 수도 있다. 모션 추정 유닛 (42) 은 루마 성분들에 대해 모션 추정을 수행할 수도 있으며, 모션 보상 유닛 (44) 은 크로마 성분들 및 루마 성분들 양쪽에 대해 루마 성분들에 기초하여 계산된 모션 벡터들을 이용할 수도 있다. 모드 선택 유닛 (40) 은 또한 비디오 슬라이스의 비디오 블록들을 디코딩할 때에 비디오 디코더 (30) 에 의한 사용을 위해 비디오 블록들 및 비디오 슬라이스와 연관되는 신택스 엘리먼트들을 발생시킬 수도 있다.

신택스 엘리먼트들은 비디오 시퀀스 레벨, 비디오 프레임 레벨, 비디오 슬라이스 레벨, 비디오 CU 레벨, 또는 비디오 PU 레벨 중 하나 이상에서의 예측 정보를 나타낼 수도 있다. 예를 들어, 모션 보상 유닛 (44) 은 CU들, PU들, 및 TU들의 사이즈를 포함한 비디오 블록 정보, 및 인트라-모드 예측을 위한 모션 벡터 정보를 나타내는 신택스 엘리먼트들을 발생시킬 수도 있다.

[0068] 인트라-예측 유닛 (46) 은 위에서 설명한 바와 같이, 모션 추정 유닛 (42) 및 모션 보상 유닛 (44) 에 의해 수행되는 인터-예측에 대한 대안으로서, 현재의 블록을 인트라-예측하거나 또는 계산할 수도 있다. 특히, 인트라-예측 유닛 (46) 은 현재의 블록을 인코딩하는데 사용할 인트라-예측 모드를 결정할 수도 있다. 일부 예들에서, 인트라-예측 유닛 (46) 은 예컨대, 별개의 인코딩 패스들 동안 여러 인트라-예측 모드들을 이용하여 현재의 블록을 인코딩할 수도 있으며, 인트라-예측 유닛 (46) (또는, 일부 예들에서는, 모드 선택 유닛 (40)) 은 테스트된 모드들 중에서 사용할 적합한 인트라-예측 모드를 선택할 수도 있다.

[0069] 예를 들어, 인트라-예측 유닛 (46) 은 여러 테스트된 인트라-예측 모드들에 대한 레이트-왜곡 분석을 이용하여 레이트-왜곡 값들을 계산하고, 그 테스트된 모드들 중에서 최상의 레이트-왜곡 특성들을 갖는 인트라-예측 모드를 선택할 수도 있다. 레이트-왜곡 분석은 일반적으로 인코딩된 블록과 그 인코딩된 블록을 발생하기 위해 인코딩되었던 원래의 미인코딩된 블록 사이의 왜곡의 양 (또는, 에러) 뿐만 아니라, 그 인코딩된 블록을 발생하는데 사용된 비트레이트 (즉, 비트들의 수) 를 결정한다. 인트라-예측 유닛 (46) 은 여러 인코딩된 블록들

에 대한 왜곡들 및 레이트들로부터 비율들을 계산하여, 어느 인트라-예측 모드가 그 블록에 대해 최상의 레이트-왜곡 값을 나타내는 지를 결정할 수도 있다.

[0070] 블록에 대한 인트라-예측 모드를 선택한 후, 인트라-예측 유닛 (46) 은 블록에 대한 그 선택된 인트라-예측 모드를 나타내는 정보를 엔트로피 인코딩 유닛 (56) 에 제공할 수도 있다. 엔트로피 인코딩 유닛 (56) 은 그 선택된 인트라-예측 모드를 나타내는 정보를 인코딩할 수도 있다. 비디오 인코더 (20) 는 복수의 인트라-예측 모드 인덱스 테이블들 및 복수의 수정된 인트라-예측 모드 인덱스 테이블들 (또한, 코드워드 맵핑 테이블들로서 지칭됨) 을 포함할 수도 있는 그 송신되는 비트스트림 구성 데이터에, 여러 블록들에 대한 인코딩 컨텍스트들의 정의들, 및 가장 가능성있는 인트라-예측 모드, 인트라-예측 모드 인덱스 테이블 및 컨텍스트들의 각각에 사용할 수정된 인트라-예측 모드 인덱스 테이블의 표시들을 포함할 수도 있다.

[0071] 비디오 인코더 (20) 는 코딩중인 원래 비디오 블록으로부터, 모드 선택 유닛 (40) 으로부터의 예측 데이터를 감산함으로써 잔차 비디오 블록을 형성한다. 합산기 (50) 는 이 감산 동작을 수행할 수도 있다. 변환 프로세싱 유닛 (52) 은 이산 코사인 변환 (DCT) 또는 개념적으로 유사한 변환과 같은 변환을 잔차 블록에 적용하여, 잔차 변환 계수 값들을 포함하는 비디오 블록을 생성할 수도 있다. 변환 프로세싱 유닛 (52) 은 DCT 와 개념적으로 유사한 다른 변환들을 수행할 수도 있다. 웨이블릿 변환들, 정수 변환들, 서브밴드 변환들 또는 다른 유형들의 변환들이 또한 이용될 수도 있다. 변환 프로세싱 유닛 (52) 은 그후 그 변환을 잔차 블록에 적용하여, 잔차 변환 계수들의 블록을 발생할 수도 있다. 변환 프로세싱 유닛 (52) 은 잔차 정보를 픽셀 값 도메인으로부터 주파수 도메인과 같은 변환 도메인으로 변환할 수도 있다. 좀더 구체적으로, 변환의 적용 전에, TU 는 픽셀 도메인에서의 잔차 비디오 데이터를 포함할 수도 있으며, 변환의 적용 이후, TU 는 주파수 도메인에서의 잔차 비디오 데이터를 나타내는 변환 계수들을 포함할 수도 있다.

[0072] 종래, 비디오 인코더 (20) 는 구현된 비디오 압축 표준에 의해 지원되는 TU들의 상이한 사이즈들의 각각에 대해 별개의 컨텍스트 모델들을 유지한다. HEVC 표준에 있어, 추가적인 변환 유닛 사이즈들, 예컨대, 32x32 내지 최고 128x128 이, 비디오 코딩 효율을 향상시키기 위해 사용될 수도 있지만, 추가적인 TU 사이즈들은 또한 추가적인 변환 유닛 사이즈들의 각각에 대한 컨텍스트 모델들을 유지하기 위해 증가된 메모리 및 계산 요구사항들을 초래한다. 일부의 경우, 더 큰 TU 사이즈들은 더 많은 컨텍스트들을 이용할 수도 있으며, 이것은 더 큰 TU 사이즈들에 대해 증가된 개수의 컨텍스트들을 유지하기 위해 증가된 메모리 및 계산 요구사항을 초래할 수도 있다. 이 문제의 영향들을 감소시키기 위해, 변환 프로세싱 유닛 (52) 은 변환 (종종 "순방향 변환" 으로서 지칭됨) 을 단순화하고 매트릭스 곱셈 동안 그의 비트 심도를 제한하는 것 (예컨대, 아래에서, 그리고 도 4 및 도 5 와 관련하여 설명되는 내부 비트 심도 제한 방법들) 에 대해 위에서 그리고 아래에서 설명되는 방법들 중 임의의 방법을 수행하도록 더 구성될 수도 있다.

[0073] 변환 프로세싱 유닛 (52) 은 결과적인 변환 계수들을 양자화 유닛 (54) 으로 전송할 수도 있다. 양자화 유닛 (54) 은 그후 변환 계수들을 양자화하여, 비트 레이트를 추가로 감소시킬 수도 있다. 양자화 프로세스는 그 계수들의 일부 또는 모두와 연관되는 비트 심도를 감소시킬 수도 있다. 양자화의 정도는 양자화 파라미터를 조정함으로써 변경될 수도 있다. 일부 예들에서, 양자화 유닛 (54) 은 그후 양자화된 변환 계수들을 포함하는 매트릭스의 스캐닝을 수행할 수도 있다. 이의 대안으로, 엔트로피 인코딩 유닛 (56) 이 그 스캐닝을 수행할 수도 있다.

[0074] 일 예로서, 변환 프로세싱 유닛 (52) 은 변환 결과들을 미리 결정된 비트 심도 값 (예컨대, 16-비트 비트 심도 또는 다른 비트 심도 값) 으로 제한할 수도 있다. 일 구현예에서, 변환 프로세싱 유닛 (52) 은 하나 이상의 내부 변환 스테이지들에서 변환 결과들을 제한할 수도 있다. 비디오 인코더 (20) 는 그후 미리 결정된 비트 심도 값에 최적화된 특수화된 명령 세트들을 이용할 수도 있다. 이러한 방법으로, 변환 프로세싱 유닛 (52) 은 더 빠른 프로세싱 속도들을 겪을 수도 있다. 이 프로세스는 도 4 에서 추가로 설명되고 도시된다.

[0075] 위에서 설명된 예에서, 변환 프로세싱 유닛 (52) 은 변환 결과들을 16-비트 값들로 제한하도록 구성된다. 다른 경우, 변환 프로세싱 유닛 (52) 은 그 제한된 값들이 만족스러운 결과들을 산출할 것이라는 것을 결정하도록 더 구성될 수도 있다. 이 프로세스는 도 5 에서 추가로 설명되고 도시된다.

[0076] 양자화 이후, 엔트로피 인코딩 유닛 (56) 은 양자화된 변환 계수들을 엔트로피 코딩할 수도 있다. 예를 들어, 엔트로피 인코딩 유닛 (56) 은 컨텍스트 적응 가변 길이 코딩 (CAVLC), 컨텍스트 적응 2진 산술 코딩 (CABAC), 신택스-기반 컨텍스트-적응 2진 산술 코딩 (SBAC), 확률 간격 파티셔닝 엔트로피 (PIPE) 코딩 또는 또 다른 엔트로피 인코딩 기법을 수행할 수도 있다. 컨텍스트-기반의 엔트로피 인코딩의 경우, 컨텍스트는 이웃하는 블록들에 기초할 수도 있다. 엔트로피 인코딩 유닛 (56) 에 의한 엔트로피 인코딩 이후, 인코딩된

비트스트림은 또 다른 디바이스 (예컨대, 비디오 디코더 (30)) 로 송신되거나 또는 추후 송신 또는 취출을 위해 아카이브될 수도 있다.

[0077] 역양자화 유닛 (58) 및 역변환 유닛 (60) 은 역양자화 및 역변환을 각각 적용하여, 예컨대, 참조 블록으로 추후 사용을 위해, 픽셀 도메인에서 잔차 블록을 복원할 수도 있다. 모션 보상 유닛 (44) 은 잔차 블록을 참조 프레임 메모리 (64) 의 프레임들 중 하나의 예측 블록에 가산함으로써 참조 블록을 계산할 수도 있다. 모션 보상 유닛 (44) 은 또한 하나 이상의 내삽 필터들을 그 복원된 잔차 블록에 적용하여, 모션 추정에 사용하기 위한 서브-정수 픽셀 값들을 계산할 수도 있다. 합산기 (62) 는 복원된 잔차 블록을 모션 보상 유닛 (44) 에 의해 발생된 모션 보상된 예측 블록에 가산하여, 참조 프레임 메모리 (64) 에의 저장을 위해, 복원된 비디오 블록을 발생할 수도 있다. 복원된 비디오 블록은 그후 후속 비디오 프레임에서 블록을 인터-코딩하기 위해 모션 추정 유닛 (42) 및 모션 보상 유닛 (44) 에 의해 참조 블록으로서 사용될 수도 있다.

[0078] 도 3 은 본 개시물에서 설명하는 양태들에 따른 기법들을 구현할 수도 있는 예시적인 비디오 디코더의 예를 예시하는 블록도이다. 본 개시물에서 설명하는 기법들은 비디오 디코더 (30) 의 여러 구성요소들을 이용할 수도 있다. 일부 예들에서, 프로세서 (미도시) 는 기법들 중 임의의 기법 또는 모두를 수행하도록 구성될 수도 있다.

[0079] 도 3 의 예에서, 비디오 디코더 (30) 는 엔트로피 디코딩 유닛 (70), 모션 보상 유닛 (72) 과 인트라-예측 유닛 (74) 을 더 포함하는 예측 유닛 (81), 역양자화 유닛 (76), 역변환 유닛 (78), 참조 픽처 (프레임) 메모리 (82) 및 합산기 (80) 를 포함한다. 비디오 디코더 (30) 는 비디오 인코더 (20) (예컨대, 도 1 및 도 2 참조) 에 대해 설명되는 인코딩 패스와는 일반적으로 반대인 디코딩 패스를 수행할 수도 있다. 모션 보상 유닛 (72) 은 엔트로피 디코딩 유닛 (70) 으로부터 수신된 모션 벡터들에 기초하여 예측 데이터를 발생시킬 수도 있는 반면, 인트라 예측 유닛 (74) 은 엔트로피 디코딩 유닛 (70) 으로부터 수신된 인트라-예측 모드 표시자들에 기초하여 예측 데이터를 발생시킬 수도 있다.

[0080] 종래, 비디오 디코더 (30) 는 구현된 비디오 압축 표준에 의해 지원되는 TU들의 상이한 사이즈들의 각각에 대해 별개의 컨텍스트 모델들을 유지하였었다. HEVC 표준에 있어서, 추가적인 변환 유닛 사이즈들, 예컨대, 32x32 내지 최고 128x128 이, 비디오 코딩 효율을 향상시키는데 이용될 수도 있지만, 추가적인 TU 사이즈들은 또한 추가적인 변환 유닛 사이즈들의 각각에 대해 컨텍스트 모델들을 유지하기 위해 증가된 메모리 및 계산 요구사항들을 초래한다.

[0081] 디코딩 프로세스 동안, 비디오 디코더 (30) 는 인코딩된 비디오 슬라이스의 비디오 블록들 및 연관되는 신택스 엘리먼트들을 나타내는 인코딩된 비디오 비트스트림을 비디오 인코더 (20) 로부터 수신할 수도 있다. 비디오 디코더 (30) 의 엔트로피 디코딩 유닛 (70) 은 그 비트스트림을 엔트로피 디코딩하여, 양자화된 계수들, 모션 벡터들 또는 인트라-예측 모드 표시자들, 및 다른 신택스 엘리먼트들을 발생한다. 엔트로피 디코딩 유닛 (70) 은 그후 모션 벡터들, 및 다른 신택스 엘리먼트들을 모션 보상 유닛 (72) 으로 포워딩할 수도 있다. 비디오 디코더 (30) 는 신택스 엘리먼트들을 비디오 슬라이스 레벨 및/또는 비디오 블록 레벨에서 수신할 수도 있다.

[0082] 비디오 슬라이스가 인트라-코딩된 (I) 슬라이스로서 코딩될 때, 인트라 예측 유닛 (74) 은 시그널링된 인트라 예측 모드 및 현재의 프레임 또는 픽처의 이전에 디코딩된 블록들로부터의 데이터에 기초하여, 현재의 비디오 슬라이스의 비디오 블록에 대한 예측 데이터를 발생시킬 수도 있다. 비디오 프레임이 인터-코딩된 (예컨대, B, P 또는 GPB) 슬라이스로서 코딩될 때, 모션 보상 유닛 (72) 은 엔트로피 디코딩 유닛 (70) 으로부터 수신된 모션 벡터들 및 다른 신택스 엘리먼트들에 기초하여 현재의 비디오 슬라이스의 비디오 블록에 대한 예측 블록들을 발생할 수도 있다. 예측 블록들은 참조 픽처 리스트들 중 하나 내 참조 픽처들 중 하나로부터 발생될 수도 있다. 비디오 디코더 (30) 는, 참조 픽처 (프레임) 메모리 (82) 에 저장된 참조 픽처들에 기초하여 디폴트 구성 기법들을 이용하여, 참조 프레임 리스트들, 즉, List 0 및 List 1 을 구성할 수도 있다. 모션 보상 유닛 (72) 은 모션 벡터들 및 다른 신택스 엘리먼트들을 파싱하여 현재의 비디오 슬라이스의 비디오 블록에 대한 예측 정보를 결정하고, 그리고, 그 예측 정보를 이용하여, 디코딩중인 현재의 비디오 블록에 대한 예측 블록들을 생성할 수도 있다. 예를 들어, 모션 보상 유닛 (72) 은 그 수신된 신택스 엘리먼트들 중 일부를 이용하여, 비디오 슬라이스의 비디오 블록들을 코딩하는데 사용되는 예측 모드 (예컨대, 인트라- 또는 인터-예측), 인터-예측 슬라이스 유형 (예컨대, B 슬라이스, P 슬라이스, 또는 GPB 슬라이스), 슬라이스에 대한 참조 픽처 리스트들 중 하나 이상에 대한 구성 정보, 슬라이스의 각각의 인터-인코딩된 비디오 블록에 대한 모션 벡터들, 슬라이스의 각각의 인터-코딩된 비디오 블록에 대한 인터-예측 상태, 및 다른 정보를 결정하여, 현재의 비디오

슬라이스에서의 비디오 블록들을 디코딩할 수도 있다.

[0083] 모션 보상 유닛 (72) 은 또한 내삽 필터들에 기초하여 내삽을 수행할 수도 있다. 모션 보상 유닛 (72) 은 비디오 블록들의 인코딩 동안 비디오 인코더 (20) 에 의해 사용되는 것과 같은 내삽 필터들을 이용하여, 참조 블록들의 서브-정수 픽셀들에 대한 내삽된 값들을 계산할 수도 있다. 이 경우, 모션 보상 유닛 (72) 은 수신된 신택스 엘리먼트들로부터 비디오 인코더 (20) 에 의해 사용되는 내삽 필터들을 결정하고 그 내삽 필터들을 이용하여 예측 블록들을 발생할 수도 있다.

[0084] 역양자화 유닛 (76) 은 비트스트림으로 제공되어 엔트로피 디코딩 유닛 (70) 에 의해 디코딩되는 양자화된 변환 계수들을 역양자화, 예컨대, 양자화 해제할 수도 있다. 역양자화 프로세스는 양자화의 정도를 결정하기 위해, 그리고, 이와 유사하게, 적용되어야 하는 역양자화의 정도를 결정하기 위해, 비디오 슬라이스에서의 각각의 비디오 블록에 대한, 비디오 디코더 (30) 에 의해 계산된 양자화 파라미터 QPY 의 사용을 포함할 수도 있다.

[0085] 역변환 유닛 (78) 은 픽셀 도메인에서 잔차 블록들을 발생하기 위해, 예를 들어, 역 DCT, 역 정수 변환, 또는 개념적으로 유사한 역변환 프로세스와 같은 역변환을 변환 계수들에 적용할 수도 있다. 모션 보상 유닛 (72) 이 모션 벡터들 및 다른 신택스 엘리먼트들에 기초하여 현재의 비디오 블록에 대한 예측 블록을 발생한 후, 비디오 디코더 (30) 는 역변환 유닛 (78) 으로부터의 잔차 블록들을 모션 보상 유닛 (72) 에 의해 발생된 대응하는 예측 블록들과 합산함으로써, 디코딩된 비디오 블록을 형성할 수도 있다. 합산기 (80) 는 이 합산 동작을 수행할 수도 있다. 블록킹 현상 아티팩트들을 제거하기 위해 디블록킹 필터가 또한 그 디코딩된 블록들을 필터링하는데 적용될 수도 있다. (코딩 루프 중에 또는 코딩 루프 이후에) 다른 루프 필터들이 또한 픽셀 전환들 (pixel transitions) 을 평활화하거나 또는 아니면 비디오 품질을 향상시키기 위해 사용될 수도 있다. 주어진 프레임 또는 픽처에서 디코딩된 비디오 블록들은 그후 참조 픽처 (프레임) 메모리 (82) 에 저장될 수도 있으며, 이 메모리는 후속 모션 보상을 위해 사용되는 참조 픽처들을 저장할 수도 있다. 참조 픽처 (프레임) 메모리 (82) 는 또한 도 1 의 디스플레이 디바이스 (32) 와 같은 디스플레이 디바이스 상에의 추후 프리젠테이션을 위해, 디코딩된 비디오를 저장할 수도 있다.

[0086] 도 4 는 예를 들어, 메시-기반의 계산들을 이용한, 저 복잡성 순방향 변환의 방법 (400) 의 플로우차트를 예시한다. 방법 (400) 은 프로세서 또는 인코더, 예컨대, 예를 들어, 도 2 의 인코더 (20) 에 의해 수행될 수도 있다. 일 실시형태에서, 도 2 의 인코더 (20) 의 변환 프로세싱 유닛 (52) 은 방법 (400) 을 수행하도록 구성된다. 방법들의 여러 블록들은 변환 프로세싱 유닛에 의해 수행되는 것으로 설명되지만, 방법들의 여러 블록은 다른 프로세서들, 인코더들, 또는 그의 유닛들에 의해 수행될 수도 있는 것으로 이해되어야 한다.

[0087] 방법 (400) 은 인코더로 하여금, (예컨대, ARM 아키텍처, 진보된 SIMD (NEON), 디지털 신호 프로세싱 (DSP), 등) 16-비트 데이터에 대해 최적화된 특수화된 명령 세트들을 이용하여 16-비트 동작들을 수행하는 것을 가능하게 하기 위해서, 변환 결과들을 16-비트 값들로 제한가능하게 한다. 위에서 설명된 바와 같이, 비트 심도를 제한하는 것은 비디오 인코딩 동안 코딩 효율을 향상시키고 계산 리소스 요구사항들을 감소시킬 수도 있다. 그러나, 일부의 경우, 비트 심도를 제한하는 것은 또한 변환 출력의 품질을 감소시킬 수도 있다. 따라서, 16-비트 동작들을 수행하기 전에, 인코더 (예컨대, 인코더 (20) 의 변환 프로세싱 유닛 (52)) 는 그 제한된 값들이 만족스러운 결과들을 산출할 것이라는 것을 결정하고 그에 따라서 조정하도록 더 구성될 수도 있으며, 이는 도 5 와 관련하여 추가로 설명된다.

[0088] 방법 (400) 은 블록 (405) 에서 시작한다. 그후 블록 (410) 에서, 변환 프로세싱 유닛 (52) 은 풀-사이즈 순방향 변환 매트릭스를 다수의 덜 복잡한 스테이지들로 분해할 (예컨대, 멀티-스테이지 변환 매트릭스들을 결정할) 수도 있다. 분해는 메시-기반의 분해 방법을 이용함으로써 일어날 수도 있다. 일 구현예에서, 풀-사이즈 변환 매트릭스를 분해하기 위해 계산 리소스들을 이용하는 대신, 변환 프로세싱 유닛 (52) 은 대신에, 메모리 또는 코딩된 비트스트림으로부터 다수의 스테이지들을 취출할 수도 있다. 일부 구현예들에서, 초기 순방향 변환 매트릭스는 32x32 변환 매트릭스일 수도 있다. 다른 구현예들에서, 변환 매트릭스는 다른 사이즈일 수도 있다.

[0089] 그후 블록 (415) 에서, 변환 프로세싱 유닛 (52) 은 다수의 스테이지들의 각각 상에서 적합한 변환 동작들을 수행할 수도 있다. 일부의 경우, 변환 동작들이 수행된 후, 스테이지들의 하나 이상은 16-비트보다 더 큰 변환 결과들을 포함할 수도 있다. 이들 변환 결과들은 변환 프로세싱 유닛 (52) 의 효율을 감소시킬 수도 있는, 16-비트보다 큰 시스템들에 대해 설계된 내부 계산 동작들 (예컨대, 32-비트 동작들) 을 필요로 할 수도 있다.

[0090] 따라서, 블록 (420) 에서, 변환 프로세싱 유닛 (52) 은 다수의 스테이지들의 각각에서의 변환 결과들의 내부 비트 심도를 16-비트 값들로 제한할 수도 있다. 변환 프로세싱 유닛 (52) 은 예컨대, 내부 값들을 16-비트 값들로 클리핑함으로써, 포화 로직을 이용하여 이 제한 프로세스를 수행할 수도 있다. 좀더 구체적으로, 변환 프로세싱 유닛 (52) 은 단지 변환 출력 비트 심도를 제한하기 보다는, 변환 동작 동안 내부적으로 비트 심도를 제한할 수도 있다. 이를 달성하기 위해, 변환 프로세싱 유닛 (52) 은 각각의 중간 변환 스테이지의 출력에서 클리핑 함수를 수행할 수도 있다. 일 구현예에서, 변환 프로세싱 유닛 (52) 은 부록 A 에 예시된 코드에 첨부된 클리핑 함수를 이용하여, 제한 프로세스를 수행할 수도 있으며, 여기서, a 및 b 는 가산될 값들을 나타내며, 그 합계는 16-비트 범위 이내로 제한된다. 하나의 예시적인 클리핑 함수는 $y = CLIP_3(x, min_{val}, max_{val})$ 로서 표현될 수도 있다. 이 예시적인 함수에서, x 는 바이트들의 수를 나타낼 수도 있다. x 가 min_{val} 미만이면, y 는 min_{val} 로 설정될 수도 있다. 또, x 가 max_{val} 보다 더 크면, y 는 max_{val} 으로 설정될 수도 있다. x 가 min_{val} 과 max_{val} 의 범위 사이에 들어가면, y 는 x 로 설정될 수도 있다. 좀더 구체적인 예시적인 클리핑 함수는 상기 클리핑 예와 유사한 로직 트리들을 따르는 $(a+b)_{sat} = CLIP_3(-32768, 32767, a+b)$ 또는 $(a_0 * b_0 + a_1 * b_1 + \cdots)_{sat} = CLIP_3(-32768, 32767, a_0 * b_0 + a_1 * b_1 + \cdots)$ 중 하나로 표현될 수도 있다. 클리핑 함수들의 다른 구조들이 제한 프로세스를 수행하기 위해 구현될 수도 있다. 어쨌든, 가산, 감산, 및/또는 곱셈 및 가산 동작들을 위해, 포화 로직은 결과들을 16-비트로 제한할 수도 있다.

[0091] 각각의 스테이지에서 내부 비트 심도를 16-비트로 제한하는 것은 변환 프로세싱 유닛 (52) 으로 하여금 그 레벨에 대해 계산 효율적인 명령 세트들을 이용가능하게 할 수도 있다. 예를 들어, 변환을 위한 입력 비트 심도는 9-비트일 수도 있으며 시작 내부 비트 심도는 16-비트보다 더 클 수도 있다. 일 예에서, 변환 프로세싱 유닛 (52) 은 단지 특정의 비트-레벨들에서의 동작들 (예컨대, 16-비트 동작들, 32-비트 동작들, 또는 64-비트 동작들, 등) 을 수행하는 것이 가능할 수도 있다. 따라서, 내부 비트 심도가 16-비트보다 클 때, 변환 프로세싱 유닛 (52) 은 적어도 32-비트 레벨에서 내부 동작들을 수행하도록 요구될 수도 있다. 이러한 결과를 피하기 위해, 변환 프로세싱 유닛 (52) 은 (예컨대, 32-비트 내부 동작들이 요구되지 않도록) 변환의 각각의 분해된 스테이지의 내부 비트 심도를 16-비트로 제한할 수도 있다. 다시 말해서, 각각의 스테이지의 내부 비트 심도가 16-비트로 제한된 상태에서, 프로세서는 (예컨대, ARM 아키텍처, 진보된 SIMD (NEON), 디지털 신호 프로세싱 (DSP), 등) 16-비트 동작들과 함께 사용하기 위해 특별히 설계된 계산 효율적인 명령 세트들을 이용할 수도 있다.

[0092] 그후, 옵션적인 블록 (425) 에서, 변환 프로세싱 유닛 (52) 은 옵션적으로 각각의 스테이지에서의 제한된 값들 편차 (예컨대, 에러) 를 임계치와 비교함으로써 그 제한된 값들이 만족스러운 결과들을 산출하는지 여부를 결정할 수도 있다. 예를 들어, 본 방법 (500) 은 블록 (425) 에서, 도 5 와 관련하여 아래에서 설명되는 방법 (500) 의 블록들 (510 및 520) 을 수행함으로써, 제한된 값들이 만족스러운 결과들을 산출하는지 여부를 결정할 수도 있다. 제한된 값들이 만족스러운 결과들을 산출하면, 본 방법 (400) 은 블록 (430) 으로 진행한다. 그렇지 않으면, 블록 (427) 에서, 변환 프로세싱 유닛 (52) 은 (예컨대, 방법 (500) 의 블록 (530) 에 대해 아래에서 설명되는 바와 같이) 편차가 임계치보다 낮아질 때까지 계수들의 서브세트를 재계산할 수도 있다. 이 방법 (500) 은 변환 프로세싱 유닛 (52) 으로 하여금, 코딩 효율을 증가시키면서 또한 비디오 품질을 유지가능하게 할 수도 있다. 일단 제한된 값들이 만족스러운 결과들을 산출할 것이라는 것을 변환 프로세싱 유닛 (52) 이 결정하였으면, 그후 블록 (430) 에서, 변환 프로세싱 유닛 (52) 은 변환 데이터에 대해 (예컨대, ARM 아키텍처, 진보된 SIMD (NEON), 디지털 신호 프로세싱 (DSP), 등) 16-비트 데이터에 최적화된 특수화된 명령 세트들을 이용하여 16-비트 동작들을 수행하는 것으로 진행할 수도 있다. 블록 (490) 에서, 본 방법이 종료한다.

[0093] 도 5 는 저 복잡성 순방향 변환을 위한 방법 (500) 의 플로우차트를 예시한다. 방법 (500) 은 프로세서 또는 인코더, 예컨대, 도 2 와 관련하여 위에서 설명된 인코더 (20) 에 의해 수행될 수도 있다. 일 실시형태에서, 인코더의 변환 프로세싱 유닛 (예컨대, 도 2 의 인코더 (20) 의 변환 프로세싱 유닛 (52)) 은 방법 (500) 을 수행하는데 사용될 수도 있다. 실제로, 방법 (500) 은 인코더 (20) 의 변환 프로세싱 유닛 (52) 에 의해 수행되는 것으로 설명되지만, 방법 (500) 은 상이한 프로세서, 인코더, 또는 인코더의 프로세싱 유닛에 의해 수행될 수도 있는 것으로 이해되어야 한다. 일 실시형태에서, 본 방법 (500) 은 제한 값들 (예컨대, 도 4 의

방법으로부터의 제한된 값들) 이 만족스러운 결과들을 산출하는지 여부를 먼저 결정하고 그에 따라서 조정한다.

도 4 에 관해서 위에서 설명된 바와 같이, 변환 프로세싱 유닛 (52) 이 내부 비트 심도를 제한할 때, 그것이 또한 변환 출력의 품질을 감소시킬 수도 있다. 도 5 와 관련하여 설명된 방법들은 품질에서의 감소가 어느 범위까지 일어날 수도 있는지를 결정하고 그에 따라서, 편차가 미리 결정된 임계치 아래가 될 때까지 조정한다.

[0094] 방법 (500) 은 블록 (505) 에서 시작한다. 방법 (500) 의 초기에, 변환 프로세싱 유닛 (52) 은 풀-사이즈 순방향 변환 매트릭스를 다수의, 덜 복잡한 스테이지들로 이미 분해하였다. 예를 들어, 순방향 변환 매트릭스는 도 4 와 관련하여 설명된 방법들에 따라서 분해되었을 수도 있다. 각각의 스테이지의 내부 비트 심도는 도 4 에 관하여 추가로 설명된 바와 같이, 특정의 비트 심도 값 (예컨대, 16-비트 비트 심도 값) 으로 이미 제한되었을 수도 있다.

[0095] 그후, 블록 (510) 에서, 변환 프로세싱 유닛 (52) 은 원래 값들로부터 제한된 값들 (예컨대, 최종 계수들) 의 편차를 결정할 수도 있다. 일 예로서, 포화 로직이 사용될 때, 최종 계수 값들은 원래 값들로부터 벗어날 (예컨대, 에러의 레벨을 포함할) 수도 있다.

[0096] 편차를 계산한 후, 블록 (520) 에서, 변환 프로세싱 유닛 (52) 은 편차가 미리 결정된 임계치 한계보다 더 큰지 여부를 결정할 수도 있다. 만약 그렇다면, 방법 (500) 은 블록 (530) 으로 진행한다. 블록 (530) 에서, 변환 프로세싱 유닛 (52) 은 (예컨대, 변환을 부분적으로 적용하여 AC-레벨 계수들의 서브세트를 재계산함으로써) 계수들의 서브세트를 재계산하고 그 동일한 제한된 값들의 서브세트를 재계산된 결과들로 대체할 수도 있다. 이 프로세스는 원래 값들로부터 최종 계수 값들이 가지는 편차를 보상하면서 또한 계산 요구사항들 (예컨대, 사이클들) 및 복잡성을 최소화할 수도 있다. 일 구현예에서, 계수들의 서브세트는, DC-레벨 계수들이 AC-레벨 계수들보다 더 큰 동적 범위들을 가지며; 따라서, 그들이 원래 값들로부터 벗어날 가능성이 더 많기 때문에, 단지 DC-레벨 계수들만을 포함할 수도 있다. 다른 구현예들에서, 그리고 정확도를 더욱 증가시키기 위해, 계수들의 서브세트는 DC-레벨 계수들을 포함하는 것에 더해서, DC-레벨에 가까운 값들을 가지는 AC-레벨 계수들을 포함할 수도 있다.

[0097] 위에서 설명한 바와 같이 계수들의 작은 서브세트를 재계산한 후, 변환 프로세싱 유닛은 그후 블록 (510) 로 복귀하여, 임계치에 대해 다시 편차를 체크할 수도 있다. 일단 (블록 (520) 에서) 편차가 미리 결정된 임계치 내구 한계 미만이라고 변환 프로세싱 유닛 (52) 이 결정하였으면, 본 방법 (500) 은 블록 (590) 로 진행하여 종료한다. 블록 (590) 에서, 변환 프로세싱 유닛 (52) 은 제한된 값들이 만족스러운 결과들을 산출할 것이라고 결정하고, 재-계산된 계수들의 서브세트를 제공할 수도 있다. 변환 프로세싱 유닛 (52) 은 그후 도 4 와 관련하여 위에서 설명한 방법 (400) 의 블록 (427) 에서, 재-계산된 계수들의 서브세트를 이용한다.

[0098] 상기 방법들의 모두는 NxN 변환에 대해 설명되었다. 그러나, 본 개시물에서 설명되는 방법들은 NxN 변환들에 제한되지 않는다. 본 방법들은 또한 변환들에서의 치수들의 수에 관계없이, 임의의 사이즈 변환들에 대해 구현될 수도 있다.

[0099] 그 예에 따라서, 본원에서 설명되는 기법들 중 임의의 기법의 어떤 행위들 또는 이벤트들이 상이한 시퀀스로 수행될 수 있으며, 추가되거나, 병합되거나, 또는 모두 제외시킬 수도 있는 (예컨대, 모든 설명되는 행위들 또는 이벤트들이 기법들의 실시에 필수적인 것은 아닌) 것으로 인식되어야 한다. 더욱이, 어떤 예들에서, 행위들 또는 이벤트들은 순차적으로 보다는, 동시에, 예컨대, 멀티-쓰레드된 프로세싱, 인터럽트 프로세싱, 또는 다수의 프로세서들을 통해서 수행될 수도 있다.

[0100] 하나 이상의 예들에서, 설명된 기능들은 하드웨어, 소프트웨어, 펌웨어, 또는 이들의 임의의 조합으로 구현될 수도 있다. 소프트웨어로 구현되는 경우, 그 기능들은 하나 이상의 명령들 또는 코드로서, 컴퓨터-판독가능 매체 상에 저장되거나 또는 컴퓨터-판독가능 매체를 통해서 송신될 수도 있으며, 하드웨어-기반의 프로세싱 유닛에 의해 실행될 수도 있다. 컴퓨터 판독가능 매체들은 데이터 저장 매체들과 같은 유형의 매체에 대응하는 컴퓨터 판독가능 저장 매체들, 또는 예를 들어, 통신 프로토콜에 따라 일 장소로부터 다른 장소로의 컴퓨터 프로그램의 전송을 용이하게 하는 임의의 매체를 포함하는 통신 매체들을 포함할 수도 있다. 이런 방법으로, 컴퓨터-판독가능 매체들은 일반적으로 (1) 비일시성 유형의 컴퓨터-판독가능 저장 매체, 또는 (2) 신호 또는 캐리어 파와 같은 통신 매체에 대응할 수도 있다. 데이터 저장 매체는 본 개시물에서 설명하는 기법들의 구현을 위한 명령들, 코드 및/또는 데이터 구조들을 취출하기 위해 하나 이상의 컴퓨터들 또는 하나 이상의 프로세서들에 의해 액세스될 수 있는 임의의 가용 매체들일 수도 있다. 컴퓨터 프로그램 제품은 컴퓨터-판독가능 매체를 포함할 수도 있다.

[0101] 일 예로서, 이에 한정하지 않고, 이런 컴퓨터-판독가능 저장 매체는 RAM, ROM, EEPROM, CD-ROM 또는 다른 광디스크 스토리지, 자기디스크 스토리지, 또는 다른 자기 저장 디바이스들, 플래시 메모리, 또는 원하는 프로그램 코드를 명령들 또는 데이터 구조들의 형태로 저장하는데 사용될 수 있고 컴퓨터에 의해 액세스될 수 있는 임의의 다른 매체를 포함할 수 있다. 또한, 임의의 접속이 컴퓨터-판독가능 매체로 적절히 지칭된다. 예를 들어, 동축 케이블, 광섬유 케이블, 연선, 디지털 가입자 회선 (DSL), 또는 무선 기술들, 예컨대 적외선, 라디오, 및 마이크로파를 이용하여 명령들이 웹사이트, 서버, 또는 다른 원격 소스로부터 송신되는 경우, 동축 케이블, 광섬유 케이블, 연선, DSL, 또는 무선 기술들 예컨대 적외선, 라디오, 및 마이크로파가 그 매체의 정의에 포함된다. 그러나, 컴퓨터-판독가능 저장 매체 및 데이터 저장 매체는 접속부들, 캐리어 파들, 신호들, 또는 다른 일시성 매체를 포함하지 않고, 그 대신, 비-일시성 유형의 저장 매체로 송신되는 것으로 해석되어야 한다. 디스크 (disk) 및 디스크 (disc) 는, 본원에서 사용할 때, 컴팩트 디스크 (CD), 레이저 디스크, 광 디스크, 디지털 다기능 디스크 (DVD), 플로피 디스크 및 Blu-ray 디스크를 포함하며, 디스크들 (disks) 은 데이터를 자기적으로 보통 재생하지만, 디스크들 (discs) 은 레이저로 데이터를 광학적으로 재생한다. 앞에서 언급한 것들의 결합들이 또한 컴퓨터-판독가능 매체들의 범위 내에 포함되어야 한다.

[0102] 명령들은 하나 이상의 디지털 신호 프로세서들 (DSP들), 범용 마이크로프로세서들, 주문형 집적회로들 (ASIC들), 필드 프로그래밍가능 로직 어레이들 (FPGA들), 또는 다른 등가의 집적 또는 이산 로직 회로와 같은, 하나 이상의 프로세서들에 의해 실행될 수도 있다. 따라서, 용어 "프로세서" 는, 본원에서 사용될 때 전술한 구조 중 임의의 구조 또는 본원에서 설명하는 기법들의 구현에 적합한 임의의 다른 구조를 지칭할 수도 있다. 게다가, 일부 양태들에서, 본원에서 설명하는 기능은 인코딩 및 디코딩을 위해 구성되는 전용 하드웨어 및/또는 소프트웨어 모듈들 내에 제공되거나, 또는 결합된 코덱에 포함될 수도 있다. 또한, 이 기법들은 하나 이상의 회로들 또는 로직 엘리먼트들로 전적으로 구현될 수 있다.

[0103] 본 개시물의 기법들은 무선 핸드셋, 집적 회로 (IC) 또는 IC들의 세트 (예컨대, 칩 세트) 를 포함한, 매우 다양한 디바이스들 또는 장치들로 구현될 수도 있다. 개시한 기법들을 수행하도록 구성되는 디바이스들의 기능적 양태들을 강조하기 위해서 여러 구성요소들, 모듈들, 또는 유닛들이 본 개시물에서 설명되지만, 상이한 하드웨어 유닛들에 의한 실현을 반드시 필요로 하지는 않는다. 대신, 위에서 설명한 바와 같이, 여러 유닛들이 코덱 하드웨어 유닛에 결합되거나 또는 적합한 소프트웨어 및/또는 펌웨어와 함께, 위에서 설명한 바와 같은 하나 이상의 프로세서들을 포함한, 상호작용하는 하드웨어 유닛들의 컬렉션으로 제공될 수도 있다.

[0104] 여러 예들이 설명되었다. 이들 및 다른 예들은 다음 청구항들의 범위 이내이다.

[0105] 부록 A: 16x16 순방향 변환의 메시-기반의 구현예의 예

```
for (j=0; j<16; j++)
{
 /* E and O*/
 for (k=0;k<8;k++)
 {
  nE[k] = pSrc[k] + pSrc[15-k];
  nO[k] = pSrc[k] - pSrc[15-k];
 }
 /* EE and EO */
 for (k=0;k<4;k++)
 {
  nEE[k] = nE[k] + nE[7-k];
  nEO[k] = nE[k] - nE[7-k];
 }
 /* EEE and EEO */
 nEEE[0] = nEE[0] + nEE[3];
 nEEO[0] = nEE[0] - nEE[3];
 nEEE[1] = nEE[1] + nEE[2];
 nEEO[1] = nEE[1] - nEE[2];

 pDst[ 0 ] = (anTransCoef16[ 0][0]*nEEE[0] + anTransCoef16[ 0][1]*nEEE[1] +
   4)>>3;
 pDst[ 8 ] = (anTransCoef16[ 8][0]*nEEE[0] + anTransCoef16[ 8][1]*nEEE[1] +
   4)>>3;
 pDst[ 4 ] = (anTransCoef16[ 4][0]*nEEO[0] + anTransCoef16[ 4][1]*nEEO[1] +
   4)>>3;
 pDst[ 12] = (anTransCoef16[12][0]*nEEO[0] + anTransCoef16[12][1]*nEEO[1] +
   4)>>3;

 for (k=2;k<16;k+=4)
 {
  pDst[ k ] = (anTransCoef16[k][0]*nEO[0] + anTransCoef16[k][1]*nEO[1] +
   anTransCoef16[k][2]*nEO[2] + anTransCoef16[k][3]*nEO[3] + 4)>>3;
 }
```

[0106]

```
  for (k=1;k<16;k+=2)
  {
    pDst[ k ] = (anTransCoef16[k][0]*nO[0] + anTransCoef16[k][1]*nO[1] +
     anTransCoef16[k][2]*nO[2] + anTransCoef16[k][3]*nO[3] +
      anTransCoef16[k][4]*nO[4] + anTransCoef16[k][5]*nO[5] +
     anTransCoef16[k][6]*nO[6] + anTransCoef16[k][7]*nO[7] + 4)>>3;
  }

  pSrc += 16;
  pDst += 16;

}

pSrc = pCoef;
pDst = pRes;
for (j=0; j<16; j++)
{
  /* E and O*/
  for (k=0;k<8;k++)
  {
    nE[k] = pSrc[k*16] + pSrc[(15-k)*16];
    nO[k] = pSrc[k*16] - pSrc[(15-k)*16];
  }
  /* EE and EO */
  for (k=0;k<4;k++)
  {
    nEE[k] = nE[k] + nE[7-k];
    nEO[k] = nE[k] - nE[7-k];
  }
  /* EEE and EEO */
  nEEE[0] = nEE[0] + nEE[3];
  nEEO[0] = nEE[0] - nEE[3];
  nEEE[1] = nEE[1] + nEE[2];
  nEEO[1] = nEE[1] - nEE[2];
```

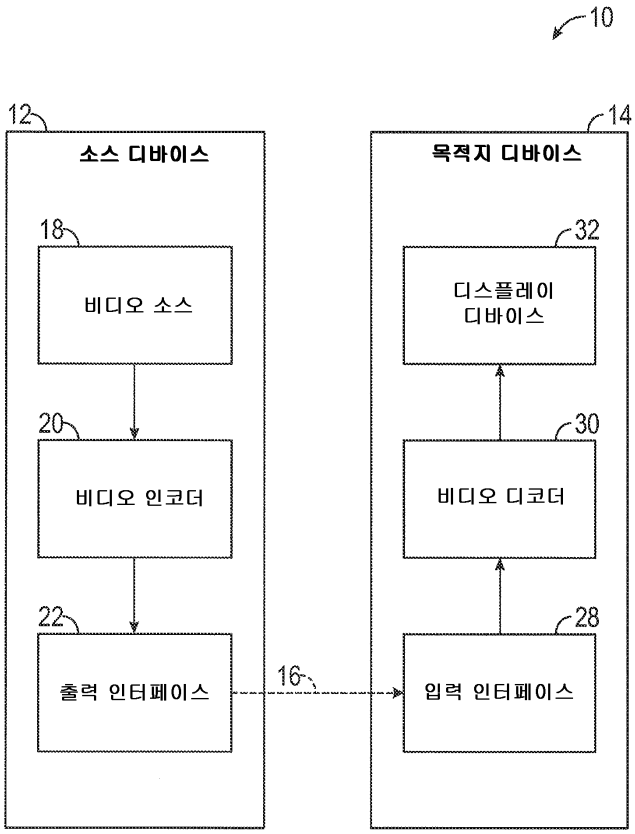
[0107]

```
pDst[ 0    ] = (anTransCoef16[ 0][0]*nEEE[0] + anTransCoef16[ 0][1]*nEEE[1] +
  512)>>10;
pDst[ 8*16 ] = (anTransCoef16[ 8][0]*nEEE[0] + anTransCoef16[ 8][1]*nEEE[1] +
  512)>>10;
pDst[ 4*16 ] = (anTransCoef16[ 4][0]*nEEO[0] + anTransCoef16[ 4][1]*nEEO[1] +
  512)>>10;
pDst[ 12*16] = (anTransCoef16[12][0]*nEEO[0] + anTransCoef16[12][1]*nEEO[1]
  + 512)>>10;

for (k=2;k<16;k+=4)
{
 pDst[ k*16 ] = (anTransCoef16[k][0]*nEO[0] + anTransCoef16[k][1]*nEO[1] +
  anTransCoef16[k][2]*nEO[2] + anTransCoef16[k][3]*nEO[3] + 512)>>10;
}

for (k=1;k<16;k+=2)
{
 pDst[ k*16 ] = (anTransCoef16[k][0]*nO[0] + anTransCoef16[k][1]*nO[1] +
  anTransCoef16[k][2]*nO[2] + anTransCoef16[k][3]*nO[3] +
   anTransCoef16[k][4]*nO[4] + anTransCoef16[k][5]*nO[5] +
  anTransCoef16[k][6]*nO[6] + anTransCoef16[k][7]*nO[7] + 512)>>10;
}

pSrc ++;
pDst ++;

}
```

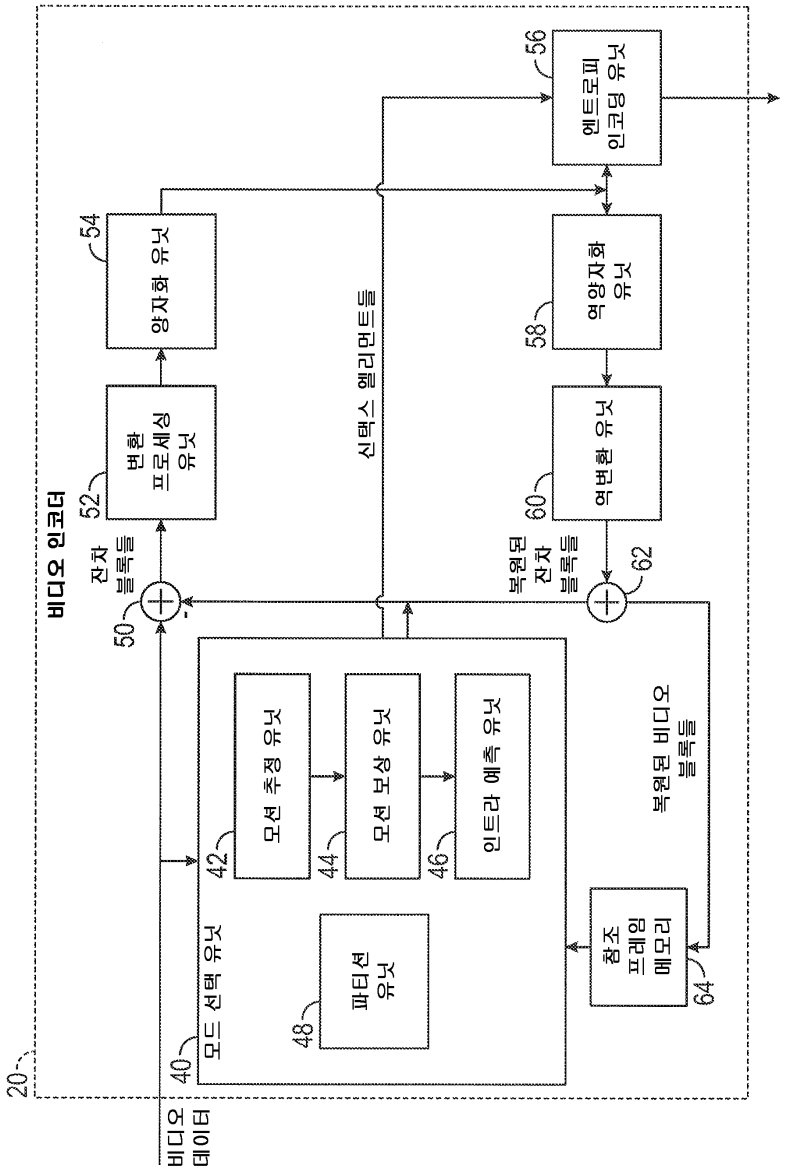
[0108]

도면

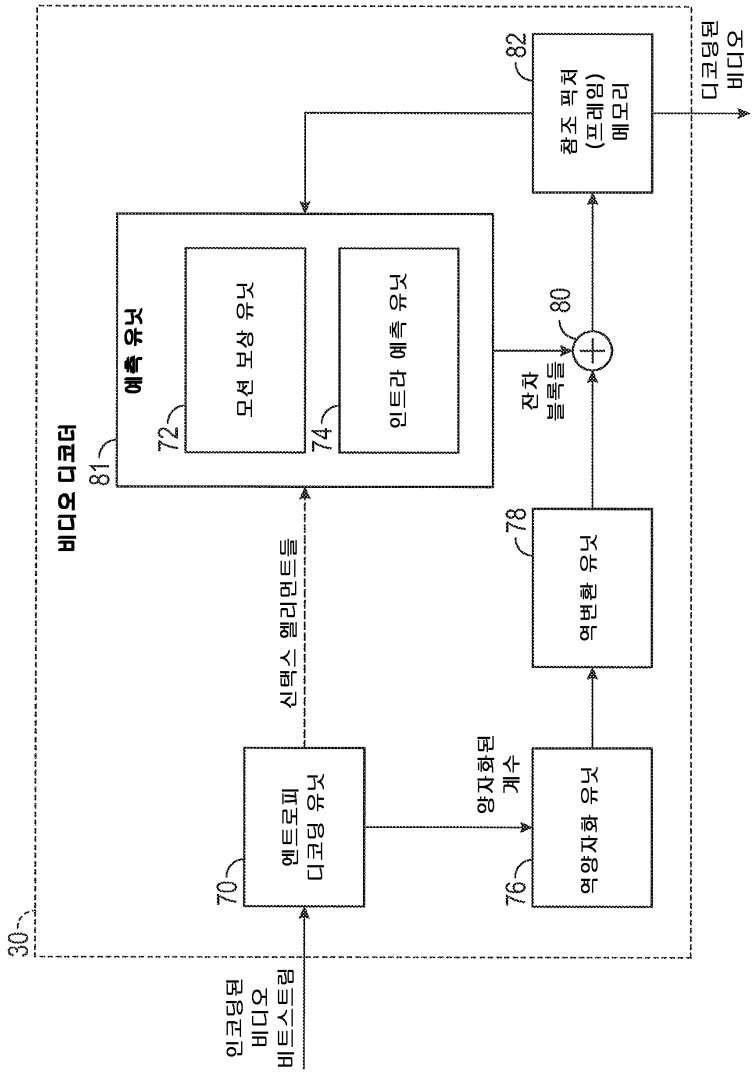
도면1


10
12
소스 디바이스
18
비디오 소스
20
비디오 인코더
22
출력 인터페이스
16
14
목적지 디바이스
32
디스플레이 디바이스
30
비디오 디코더
28
입력 인터페이스

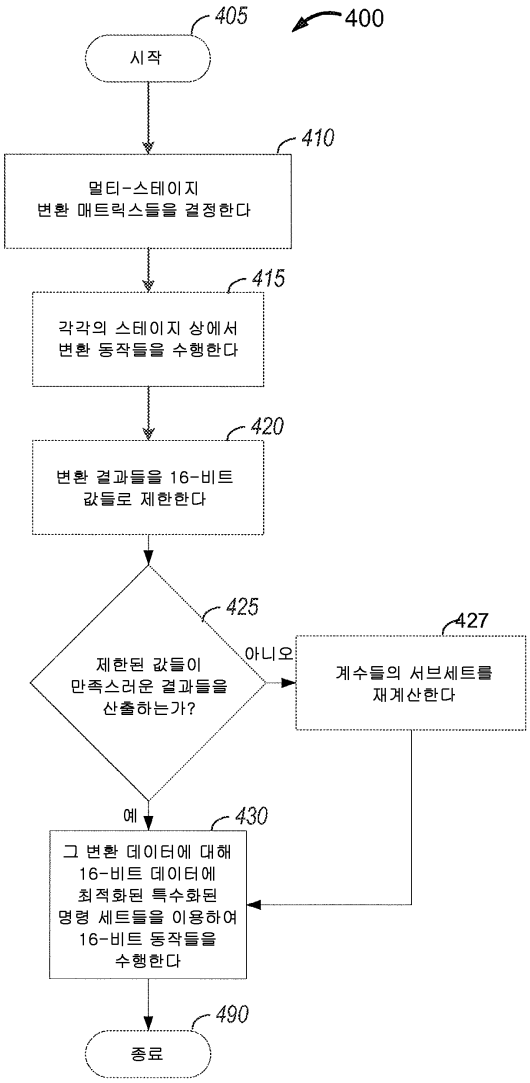
도면2


비디오 인코더
20
비디오 데이터
모드 선택 유닛
40
모션 추정 유닛
42
모션 보상 유닛
44
인트라 예측 유닛
46
파티션 유닛
48
참조 프레임 메모리
64
복원된 비디오 블록들
잔차 블록들
50
변환 프로세싱 유닛
52
양자화 유닛
54
신택스 엘리먼트들
엔트로피 인코딩 유닛
56
역양자화 유닛
58
역변환 유닛
60
복원된 잔차 블록들
62

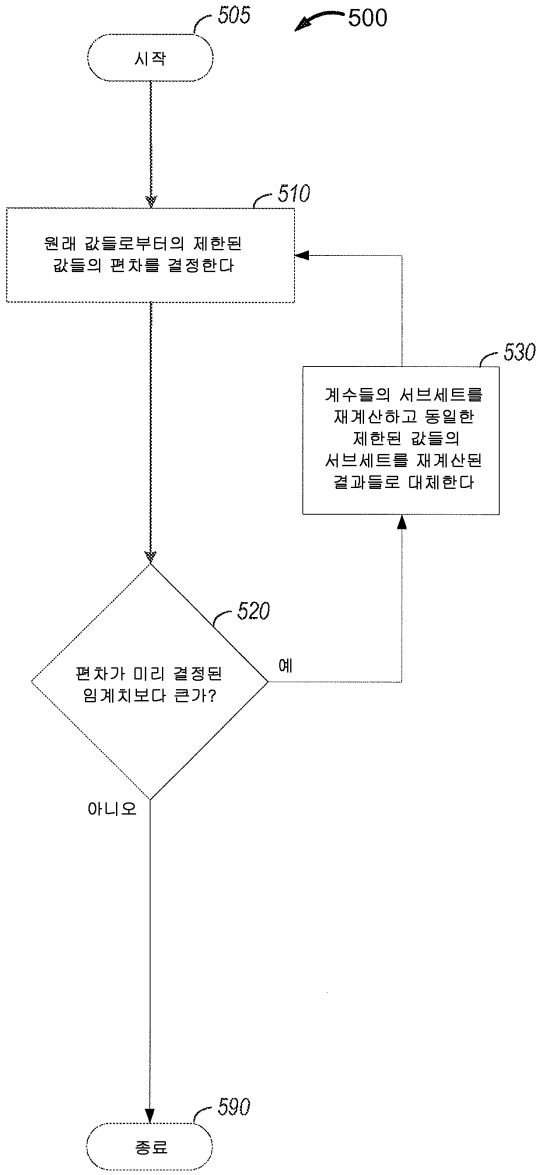
도면3


인코딩된 비디오 비트스트림
30
비디오 디코더
엔트로피 디코딩 유닛
70
신택스 엘리먼트들
예측 유닛
81
모션 보상 유닛
72
인트라 예측 유닛
74
양자화된 계수
역양자화 유닛
76
역변환 유닛
78
잔차 블록들
80
참조 픽처 (프레임) 메모리
82
디코딩된 비디오

도면4


400
405
시작
410
멀티-스테이지
변환 매트릭스들을 결정한다
415
각각의 스테이지 상에서
변환 동작들을 수행한다
420
변환 결과들을 16-비트
값들로 제한한다
425
제한된 값들이
만족스러운 결과들을
산출하는가?
아니오
427
계수들의 서브세트를
재계산한다
예
430
그 변환 데이터에 대해
16-비트 데이터에
최적화된 특수화된
명령 세트들을 이용하여
16-비트 동작들을
수행한다
490
종료

도면5


505
500
시작
510
원래 값들로부터의 제한된
값들의 편차를 결정한다
530
계수들의 서브세트를
재계산하고 동일한
제한된 값들의
서브세트를 재계산된
결과들로 대체한다
520
편차가 미리 결정된
임계치보다 큰가?
예
아니오
590
종료