(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2012-0074310
(43) 공개일자 2012년07월05일

(51) 국제특허분류(Int. Cl.)
G10L 19/00 (2006.01)
(21) 출원번호 10-2012-7012845
(22) 출원일자(국제) 2010년10월19일
심사청구일자 2012년05월18일
(85) 번역문제출일자 2012년05월18일
(86) 국제출원번호 PCT/EP2010/065725
(87) 국제공개번호 WO 2011/048098
국제공개일자 2011년04월28일
(30) 우선권주장
61/253,459 2009년10월20일 미국(US)

(71) 출원인
프라운호퍼 게젤샤프트 쭈르 푀르데룽 데어 안겐반텐 포르슝 에. 베.
독일 80686 뮌헨 한자슈트라쎄 27 체
(72) 발명자
푸쉬 귈라움
독일 에를랑겐 91058 퓌르터 스트라쎄 17
수바라만 비네쉬
독일 에를랑겐 91058 게르하르트 하우프트만 스트라쎄 11
(뒷면에 계속)
(74) 대리인
배성렬, 김원식

전체 청구항 수 : 총 18 항

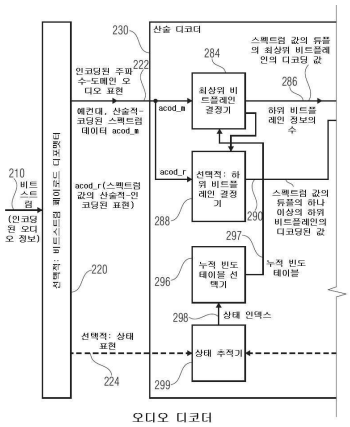
(54) 발명의 명칭 **오디오 인코더, 오디오 디코더, 오디오 정보를 인코딩하는 방법, 오디오 정보를 디코딩하는 방법 및 이전의 디코딩된 스펙트럼 값의 그룹의 검출을 이용한 컴퓨터 프로그램**

(57) 요 약

인코딩된 오디오 정보(210)에 기초하여 디코딩된 오디오 정보(212)를 제공하는 오디오 디코더(200)는 스펙트럼 값의 산술적 인코딩된 표현(222)에 기초하여 다수의 디코딩된 스펙트럼 값(232)을 제공하는 산술 디코더(230) 및, 디코딩된 오디오 정보를 획득하기 위해 디코딩된 스펙트럼 값을 이용하여 시간-도메인 오디오 표현(262)을 제공하는 주파수-도메인 대 시간-도메인 변환기(260)를 포함한다. 산술 디코더(230)는 컨텍스트 상태에 따라 심볼 코드로의 코드 값의 맵핑을 나타내는 맵핑 규칙을 선택하도록 구성된다. 산술 디코더는 다수의 이전의 디코딩된 스펙트럼 값에 따라 현재 컨텍스트 상태를 결정하거나 수정하도록 구성된다. 산술 디코더는, 개별적으로 또는 종합하여, 크기에 관해 미리 정해진 조건을 충족하는 다수의 이전의 디코딩된 스펙트럼 값의 그룹을 검출하여, 검출의 결과에 따라 현재 컨텍스트 상태를 결정하도록 구성된다.

오디오 인코더는 유사한 원리를 이용한다.

대 표 도 - 도2a



(72) 발명자

레텔바흐 니콜라우스

독일 뉘른베르그 90427 스페사르트스트라쎄 38

멀트러스 마르쿠스

독일 뉘른베르그 90469 에츠라웁베그 7

가이어 마르크

독일 에를랑겐 91058 팔켄아우어 스트라쎄 3

웜볼드 패트릭

독일 엠스키르헨 91448 마우스도르프 50

그리벨 크리스티앙

독일 뉘른베르그 90402 오스텐드스트라쎄 44

바이스 올리버

독일 뉘른베르그 90459 피터 헨라인 스트라쎄 45

특허청구의 범위

청구항 1

인코딩된 오디오 정보(210;810)에 기초하여 디코딩된 오디오 정보(212;812)를 제공하는 오디오 디코더(200;800)에 있어서,

디코딩된 스펙트럼 값의 산술적-인코딩된 표현(222;821)에 기초하여 다수의 디코딩된 스펙트럼 값(232;822)을 제공하는 산술 디코더(230;820); 및

상기 디코딩된 오디오 정보(212;812)를 획득하기 위해 상기 디코딩된 스펙트럼 값(232;822)을 이용하여 시간-도메인 오디오 표현(262;812)을 제공하는 주파수-도메인 대 시간-도메인 변환기(260;830);를 포함하고,

상기 산술 디코더(230;820)는 컨텍스트 상태(s)에 따라 심볼 코드(symbol)로의 코드 값(value)의 맵핑을 나타내는 맵핑 규칙(297; cum_freq[])을 선택하도록 구성되고,

상기 산술 디코더(230;820)는 다수의 이전의 디코딩된 스펙트럼 값에 따라 현재 컨텍스트 상태(s)를 결정하도록 구성되며,

상기 산술 디코더는, 개별적으로 또는 종합하여, 크기에 관해 미리 정해진 조건을 충족하는 다수의 이전의 디코딩된 스펙트럼 값의 그룹을 검출하여, 상기 검출의 결과에 따라 상기 현재 컨텍스트 상태(s)를 결정하거나 수정하도록 구성되는 것을 특징으로 하는 오디오 디코더.

청구항 2

청구항 1에 있어서,

상기 산술 디코더는 상기 미리 정해진 조건이 충족되는 상기 검출에 응답하여 상기 이전의 디코딩된 스펙트럼 값과 무관하게 상기 현재 컨텍스트 상태(s)를 결정하거나 수정하도록 구성되는 것을 특징으로 하는 오디오 디코더.

청구항 3

청구항 1 또는 2에 있어서,

상기 산술 디코더는, 개별적으로 또는 종합하여, 크기에 관해 미리 정해진 조건을 충족하는 다수의 이전의 디코딩된 인접 스펙트럼 값의 그룹을 검출하도록 구성되는 것을 특징으로 하는 오디오 디코더.

청구항 4

청구항 1 내지 3 중 어느 한 항에 있어서,

상기 산술 디코더(230)는, 개별적으로 또는 종합하여, 미리 정해진 임계 크기보다 작은 크기를 포함하는 다수의 이전의 디코딩된 인접 스펙트럼 값의 그룹을 검출하여, 상기 검출의 결과에 따라 상기 현재 컨텍스트 상태(s)를 결정하거나 수정하도록 구성되는 것을 특징으로 하는 오디오 디코더.

청구항 5

청구항 1 내지 4 중 어느 한 항에 있어서,

상기 산술 디코더는 상기 이전의 디코딩된 스펙트럼 값의 각각이 제로 값이 되는 다수의 이전의 디코딩된 인접 스펙트럼 값의 그룹을 검출하여, 상기 검출의 결과에 따라 상기 컨텍스트 상태(s)를 결정하거나 수정하도록 구성되는 것을 특징으로 하는 오디오 디코더.

청구항 6

청구항 1 내지 4 중 어느 한 항에 있어서,

상기 산술 디코더는 미리 정해진 임계값보다 작은 합산 값을 포함하는 다수의 이전의 디코딩된 인접 스펙트럼 값의 그룹을 검출하여, 상기 검출의 결과에 따라 현재 상태(s)를 결정하거나 수정하도록 구성되는 것을 특징

으로 하는 오디오 디코더.

청구항 7

청구항 1 내지 6 중 어느 한 항에 있어서,

상기 산술 디코더는 다수의 이전의 디코딩된 인접 스펙트럼 값의 그룹이, 개별적으로 또는 종합하여, 크기에 관해 미리 정해진 조건을 충족하는 검출에 응답하여 상기 현재 컨텍스트 상태(s)를 미리 정해진 값으로 설정하도록 구성되는 것을 특징으로 하는 오디오 디코더.

청구항 8

청구항 7에 있어서,

상기 산술 디코더(230)는 다수의 이전의 디코딩된 인접 스펙트럼 값의 그룹이, 개별적으로 또는 종합하여, 크기에 관해 미리 정해진 조건을 충족하는 검출에 응답하여 다수의 이전의 디코딩된 스펙트럼 값의 수치 값에 따라 선택적으로 상기 컨텍스트 상태(s)의 계산을 생략하도록 구성되는 것을 특징으로 하는 오디오 디코더.

청구항 9

청구항 1 내지 6 중 어느 한 항에 있어서,

상기 산술 디코더는, 상기 검출에 응답하여, 개별적으로 또는 종합하여, 크기에 관해 미리 정해진 조건을 충족하는 다수의 이전의 디코딩된 인접 스펙트럼 값의 그룹의 검출을 나타내는 값의 범위 내로 상기 현재 컨텍스트 상태(s)를 설정하도록 구성되는 것을 특징으로 하는 오디오 디코더.

청구항 10

청구항 1 내지 9 중 어느 한 항에 있어서,

상기 산술 디코더는 심볼 코드(symbol; m)를 디코딩된 스펙트럼 값(a)으로 맵핑하도록 구성되는 것을 특징으로 하는 오디오 디코더.

청구항 11

청구항 1 내지 10 중 어느 한 항에 있어서,

상기 산술 디코더는 제 1 시간-주파수 영역에 대한 이전의 디코딩된 스펙트럼 값을 평가하여, 개별적으로 또는 종합하여, 크기에 관한 상기 미리 정해진 조건을 충족하는 다수의 스펙트럼 값의 그룹을 검출하도록 구성되고,

상기 산술 디코더는 상기 미리 정해진 조건이 충족되지 않을 경우에 상기 제 1 시간-주파수 영역과 상이한 제 2 시간-주파수 영역에 대한 이전의 디코딩된 스펙트럼 값에 따라 상기 컨텍스트 상태(s)를 나타내는 수치 값을 획득하도록 구성되는 것을 특징으로 하는 오디오 디코더.

청구항 12

청구항 1 내지 11 중 어느 한 항에 있어서,

상기 산술 디코더는 상기 컨텍스트 상태(s)에 따라 맵핑 규칙(ari_cf_m[pki][9])을 선택하기 위해 하나 이상의 해시 테이블(ari_s_hash, ari_gs_hash)을 평가하도록 구성되는 것을 특징으로 하는 오디오 디코더.

청구항 13

입력 오디오 정보(110;710)에 기초하여 인코딩된 오디오 정보(112;712)를 제공하는 오디오 인코더(100;700)에 있어서,

주파수-도메인 오디오 표현(132;722)이 스펙트럼 값의 세트를 포함하도록 상기 입력 오디오 정보의 시간-도메인 표현(110;710)에 기초하여 상기 주파수-도메인 오디오 표현(132;722)을 제공하는 에너지-압축 시간-도메인-대-주파수-도메인 변환기(130;720); 및

가변 길이 코드워드(acod_m, acod_r)를 이용하여 스펙트럼 값(a), 또는 이의 사전 처리된 버전을 인코딩하도

록 구성되는 산술 인코더(170;730)를 포함하고,

상기 산술 인코더(170)는 스펙트럼 값(a), 또는 스펙트럼 값(a)의 최상위 비트플레인의 값(m)을 코드 값(acod_m)으로 맵핑하도록 구성되고,

상기 산술 인코더는 컨텍스트 상태(s)에 따라 코드 값으로의 스펙트럼 값 또는 스펙트럼 값의 최상위 비트플레인의 맵핑을 나타내는 맵핑 규칙을 선택하도록 구성되며,

상기 산술 인코더는 다수의 이전의 인코딩된 스펙트럼 값에 따라 상기 현재 컨텍스트 상태(s)를 결정하도록 구성되며,

상기 산술 인코더는, 개별적으로 또는 종합하여, 크기에 관한 미리 정해진 조건을 충족하는 다수의 이전의 인코딩된 스펙트럼 값의 그룹을 검출하여, 상기 검출의 결과에 따라 상기 현재 컨텍스트 상태(s)를 결정하거나 수정하도록 구성되는 것을 특징으로 하는 오디오 인코더.

청구항 14

청구항 13에 있어서,

상기 산술 인코더는 상기 미리 정해진 조건이 충족되는 상기 검출에 응답하여 상기 이전의 인코딩된 스펙트럼 값과 무관하게 상기 현재 컨텍스트 상태(s)를 결정하거나 수정하도록 구성되는 것을 특징으로 하는 오디오 인코더.

청구항 15

청구항 13 또는 14에 있어서,

상기 산술 인코더는, 개별적으로 또는 종합하여, 크기에 관해 미리 정해진 조건을 충족하는 다수의 이전의 인코딩된 인접 스펙트럼 값의 그룹을 검출하도록 구성되는 것을 특징으로 하는 오디오 인코더.

청구항 16

인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하는 방법에 있어서,

디코딩된 스펙트럼 값의 산술적-인코딩된 표현에 기초하여 다수의 디코딩된 스펙트럼 값을 제공하는 단계; 및

상기 디코딩된 오디오 정보를 획득하기 위해 상기 디코딩된 스펙트럼 값을 이용하여 시간-도메인 오디오 표현을 제공하는 단계;를 포함하고,

상기 다수의 디코딩된 스펙트럼 값을 제공하는 단계는 인코딩된 형식으로 스펙트럼 값 또는 스펙트럼 값의 최상위 비트-플레인을 나타내는 코드 값(acod_m; value)을, 컨텍스트 상태에 따라 디코딩된 형식으로 스펙트럼 값 또는 스펙트럼 값의 최상위 비트-플레인을 나타내는 심볼 코드(symbol)로 맵핑하는 것을 나타내는 맵핑 규칙을 선택하는 단계를 포함하며,

상기 현재 컨텍스트 상태는 다수의 이전의 디코딩된 스펙트럼 값에 따라 결정되며,

개별적으로 또는 종합하여, 크기에 관해 미리 정해진 조건을 충족하는 다수의 이전의 디코딩된 스펙트럼 값의 그룹이 검출되고, 상기 현재 컨텍스트 상태는 상기 검출의 결과에 따라 결정되거나 수정되는 것을 특징으로 하는 디코딩된 오디오 정보를 제공하는 방법.

청구항 17

입력 오디오 정보에 기초하여 인코딩된 오디오 정보를 제공하는 방법에 있어서,

주파수-도메인 오디오 표현이 스펙트럼 값의 세트를 포함하도록 에너지-압축 시간-도메인-대-주파수-도메인 변환을 이용하여 상기 입력 오디오 정보의 시간-도메인 표현에 기초하여 상기 주파수-도메인 오디오 표현을 제공하는 단계; 및

스펙트럼 값 또는 스펙트럼 값의 최상위 비트플레인의 값이 코드 값으로 맵핑되는 단계로서, 가변-길이 코드 워드를 이용하여 스펙트럼 값, 또는 이의 사전 처리된 버전을 산술적으로 인코딩하는 단계;를 포함하고,

코드 값으로의 스펙트럼 값 또는 스펙트럼 값의 최상위 비트플레인의 맵핑을 나타내는 맵핑 규칙은 컨텍스트 상태에 따라 선택되며,

현재 컨텍스트 상태는 다수의 이전의 인코딩된 인접 스펙트럼 값에 따라 결정되고,

개별적으로 또는 종합하여, 크기에 관한 미리 정해진 조건을 충족하는 다수의 이전의 인코딩된 스펙트럼 값의 그룹이 검출되고, 상기 현재 컨텍스트 상태는 상기 검출의 결과에 따라 결정되거나 수정되는 것을 특징으로 하는 인코딩된 오디오 정보를 제공하는 방법.

청구항 18

청구항 16 또는 17에 따른 방법을 수행하기 위한 컴퓨터 프로그램으로서, 상기 컴퓨터 프로그램은 컴퓨터상에서 구동되는 것인 컴퓨터 프로그램.

명 세 서

기 술 분 야

[0001] 본 발명에 따른 실시예들은 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하는 오디오 디코더, 입력 오디오 정보에 기초하여 인코딩된 오디오 정보를 제공하는 오디오 인코더, 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하는 방법, 입력 오디오 정보에 기초하여 인코딩된 오디오 정보를 제공하는 방법 및 컴퓨터 프로그램에 관한 것이다.

[0002] 본 발명에 따른 실시예들은, 예컨대, 소위 통합된-음성-및-오디오-코더(USAC)와 같이 오디오 인코더 또는 디코더에 이용될 수 있는 개선된 스펙트럼 잡음없는 코딩에 관한 것이다.

배 경 기 술

[0003] 다음에는, 본 발명의 배경이 본 발명 및 이의 이점의 이해를 용이하게 하기 위해 간략히 설명된다. 과거 10년 동안, 오디오 콘텐츠를 디지털식으로 저장하여 분배할 수 있는 가능성을 생성하는데 많은 노력이 기울어져 왔다. 이런 방식의 하나의 중요한 업적은 국제 표준 ISO/IEC 14496-3의 정의이다. 이 표준의 파트 3은 오디오 콘텐츠의 코딩 및 디코딩에 관한 것이고, 파트 3의 서브파트 4는 일반적인 오디오 코딩에 관한 것이다. ISO/IEC 14496 파트 3, 서브파트 4는 일반적인 오디오 콘텐츠의 인코딩 및 디코딩에 대한 개념을 정의한다. 게다가, 품질을 개선하고, 및/또는 필요한 비트율을 감소시키기 위해 추가적인 개선 사항이 제안되었다.

[0004] 상기 표준에서 설명된 개념에 따르면, 시간-도메인 오디오 신호는 시간-주파수 표현으로 변환된다. 시간-도메인에서 시간-주파수-도메인으로의 변환은 통상적으로 또한 시간-도메인 샘플의 "프레임"으로 명시되는 변환 블록을 이용하여 수행된다. 중복이 아티팩트(artifacts)를 효율적으로 방지하도록(또는 적어도 감소시키도록) 하기 때문에, 예컨대, 프레임의 절반만큼 시프트되는 중복 프레임을 이용하는 것이 유리한 것으로 발견되었다. 게다가, 윈도잉은 시간적으로 제한된 프레임의 이러한 처리에서 발생하는 아티팩트를 방지하기 위해 수행되어야 하는 것으로 발견되었다.

[0005] 입력 오디오 신호의 윈도잉된 부분을 시간-도메인에서 시간-주파수-도메인으로 변환함으로써, 에너지 압축은 많은 경우에 스펙트럼 값의 일부가 다수의 다른 스펙트럼 값보다 상당히 큰 크기를 포함하도록 획득된다. 따라서, 많은 경우에, 스펙트럼 값의 평균 크기보다 상당히 큰 크기를 갖는 스펙트럼 값의 비교적 작은 수가 있다. 에너지 압축을 생성시키는 시간-도메인 대 시간-주파수-도메인 변환의 통상의 예는 소위 수정된-이산-코사인-변환(MDCT)이다.

[0006] 스펙트럼 값은 종종 음향 심리학(psychoacoustic) 모델에 따라 스케일링되어 양자화됨으로써, 양자화 오류가 음향 심리학적으로 더 중요한 스펙트럼 값에 대해 비교적 작고, 음향 심리학적으로 덜 중요한 스펙트럼 값에 대해 비교적 크도록 한다. 스케일링 및 양자화된 스펙트럼 값은 이의 비트레이트 효율적인 표현을 제공하기 위해 인코딩된다.

[0007] 예컨대, 양자화된 스펙트럼 계수의 소위 허프만 코딩(Huffman coding)의 사용은 International Standard ISO/IEC 14496-3:2005(E), part 3, subpart 4에 설명되어 있다.

[0008] 그러나, 스펙트럼 값의 코딩의 품질은 필요한 비트레이트에 상당한 영향을 미치는 것으로 발견되었다. 또한, 종종 휴대용 소비자 기기에서 구현되어, 저렴하고 전력 소비가 낮은 오디오 디코더의 복잡도는 스펙트럼 값을 인코딩하기 위해 이용되는 코딩에 의존하는 것으로 발견되었다.

발명의 내용

해결하려는 과제

[0009] 이러한 상황에 비추어, 비트레이트-효율성과 자원 효율성 사이의 트레이드오프(trade-off)를 개선하기 위해 제공하는 오디오 콘텐츠의 인코딩 및 디코딩을 위한 개념에 대한 필요성이 존재한다.

과제의 해결 수단

[0010] 본 발명에 따른 실시예는 인코딩된 오디오 정보(또는 인코딩된 오디오 표현)에 기초하여 디코딩된 오디오 정보(또는 디코딩된 오디오 표현)를 제공하는 오디오 디코더를 생성한다. 오디오 디코더는 스펙트럼 값의 산술적 인코딩된 표현에 기초하여 다수의 디코딩된 스펙트럼 값을 제공하는 산술 디코더를 포함한다. 오디오 디코더는 또한 디코딩된 오디오 정보를 획득하기 위해 디코딩된 스펙트럼 값을 이용하여 시간-도메인 오디오 표현을 제공하는 주파수-도메인 대 시간-도메인 변환기를 포함한다. 산술 디코더는 컨텍스트(context) 상태에 따라 심볼 코드로의 코드 값의 맵핑(mapping)을 나타내는 맵핑 규칙을 선택하도록 구성된다. 산술 디코더는 다수의 이전의 디코딩된 스펙트럼 값에 따라 현재 컨텍스트 상태를 결정하도록 구성된다. 산술 디코더는, 개별적으로 또는 종합하여, 크기에 관해 미리 정해진 조건을 충족하는 다수의 이전의 디코딩된 스펙트럼 값의 그룹을 검출하여, 검출 결과에 따라 현재 컨텍스트 상태를 결정하거나 수정하도록 구성된다.

[0011] 본 발명에 따른 실시예는 크기에 관해 미리 정해진 조건을 충족하는 다수의 이전의 디코딩된(바람직하게는, 반드시 필요하지 않지만, 인접한) 스펙트럼 값의 그룹의 존재가 현재 컨텍스트 상태의 특히 효율적인 결정을 허용하는 연구 결과(finding)에 기초하는데, 그 이유는 이와 같은 이전의 디코딩된(바람직하게는 인접한) 스펙트럼 값의 그룹이 스펙트럼 표현 내의 특징적 세부 특징(characteristic feature)이어서, 현재 컨텍스트 상태의 결정을 용이하게 하는데에 이용될 수 있기 때문이다. 예컨대, 특히 작은 크기를 포함하는 다수의 이전의 디코딩된(바람직하게는 인접한) 스펙트럼 값의 그룹을 검출함으로써, 스펙트럼 내의 비교적 낮은 진폭의 부분을 인식하여, 현재 컨텍스트 상태를 조정(결정 또는 수정)할 수 있음으로써, (비트레이트 측면에서) 양호한 코딩 효율로 추가적 스펙트럼 값을 인코딩 및 디코딩할 수 있다. 대안적으로, 비교적 큰 진폭을 포함하는 다수의 이전의 디코딩된 인접스펙트럼 값의 그룹은 검출될 수 있고, 컨텍스트는 인코딩 및 디코딩의 효율을 증대시키기 위해 적절히 조정(결정 또는 수정)될 수 있다. 더욱이, 개별적으로 또는 종합하여, 미리 정해진 조건을 충족하는 다수의 이전의 디코딩된(바람직하게는 인접한) 스펙트럼 값의 그룹의 검출은 많은 이전의 디코딩된 스펙트럼 값이 조합되는 컨텍스트 계산보다 낮은 계산량(lower computational effort)으로 종종 실행 가능하다. 요약하면, 본 발명에 따라 상기 논의된 실시예는 단순한 컨텍스트 계산을 허용하고, 인접한 비교적 작은 스펙트럼 값의 그룹 또는 인접한 비교적 큰 스펙트럼 값의 그룹이 있는 특정 신호 성상도(signal constellations)에 대한 컨텍스트의 조정을 허용한다.

[0012] 바람직한 실시예에서, 산술 디코더는 미리 정해진 조건이 충족되는 검출에 응답하여 이전의 디코딩된 스펙트럼 값과 무관하게 현재 컨텍스트 상태를 결정하거나 수정하도록 구성된다. 따라서, 계산상 특히 효율적인 메커니즘은 컨텍스트를 나타내는 값의 유도를 위해 획득된다. 미리 정해진 조건을 충족하는 다수의 이전의 디코딩된 스펙트럼 값의 그룹의 검출이 이전의 디코딩된 스펙트럼 값의 계산상 힘든 숫자 조합(demanding numeric combination)을 필요로 하지 않는 간단한 메카니즘을 생성할 경우에 컨텍스트의 의미 있는 적응이 달성될 수 있는 것으로 발견되었다. 따라서, 계산량은 다른 접근법에 비해 감소된다. 또한, 컨텍스트 유도의 촉진(acceleration)은 검출에 의존하는 복잡한 계산 단계를 생략함으로써 달성될 수 있는데, 그 이유는 이와 같은 개념이 통상적으로 프로세서에서 실행되는 소프트웨어 구현에 비효율적이기 때문이다.

[0013] 바람직한 실시예에서, 산술 디코더는, 개별적으로 또는 종합하여, 크기에 관해 미리 정해진 조건을 충족하는 다수의 이전의 디코딩된 인접스펙트럼 값의 그룹을 검출하도록 구성된다.

[0014] 바람직한 실시예에서, 산술 디코더는, 개별적으로 또는 종합하여, 미리 정해진 임계 크기보다 작은 크기를 포함하는 다수의 이전의 디코딩된 인접스펙트럼 값의 그룹을 검출하여, 검출 결과에 따라 현재 컨텍스트 상태를 결정하도록 구성된다. 다수의 인접한 비교적 낮은 스펙트럼 값의 그룹은 이러한 상황에 잘 적응되는 컨텍스트를 선택하기 위해 이용될 수 있는 것으로 발견되었다. 인접한 비교적 작은 스펙트럼 값의 그룹이 있으면, 다음에 디코딩될 스펙트럼 값이 또한 비교적 작은 값을 포함하는 상당한 가능성이 있다. 따라서, 컨텍스트의 조정은 양호한 인코딩 효율을 제공하며, 시간이 소요되는 컨텍스트 계산을 회피하는데에 도움을 줄 수 있다.

[0015] 바람직한 실시예에서, 산술 디코더는 이전의 디코딩된 스펙트럼 값의 각각이 제로(0) 값이 되는 이전의 디코딩된 인접 스펙트럼 값의 그룹을 검출하여, 검출 결과에 따라 컨텍스트 상태를 결정하도록 구성된다. 스펙트럼 또는 시간적 마스킹 효과로 인해, 제로 값을 취하는 인접 스펙트럼 값의 그룹이 종종 있다는 것으로 발견

되었다. 설명된 실시예는 이러한 상황에 대한 효율적인 처리를 제공한다. 게다가, 제로로 양자화되는 인접 스펙트럼 값의 그룹의 존재는 다음 디코딩되는 스펙트럼 값이 제로 값 또는 마스킹 효과를 생성하는 비교적 큰 스펙트럼 값인 가능성을 높게 한다.

[0016] 바람직한 실시예에서, 산술 디코더는, 미리 정해진 임계값보다 작은 합산 값을 포함하는 다수의 이전의 디코딩된 인접 스펙트럼 값의 그룹을 검출하여, 검출 결과에 따라 컨텍스트 상태를 결정하도록 구성된다. 제로인 인접 스펙트럼 값의 그룹 이외에, 또한 평균하여 거의 제로(즉, 미리 정해진 임계값보다 작은 합산 값)인 인접 스펙트럼 값의 그룹은 컨텍스트의 적응을 위해 이용될 수 있는 스펙트럼 표현(예컨대, 오디오 콘텐츠의 시간-주파수 표현)의 특징적인 세부 특징을 구성하는 것으로 발견되었다.

[0017] 바람직한 실시예에서, 산술 디코더는 미리 정해진 조건의 검출에 응답하여 현재 컨텍스트 상태를 미리 정해진 값으로 설정하도록 구성된다. 이러한 반응은 구현하기가 매우 간단하여, 양호한 코딩 효율을 제공하는 컨텍스트의 적응을 야기하는 것으로 발견되었다.

[0018] 바람직한 실시예에서, 산술 디코더는 선택적으로 미리 정해진 조건의 검출에 응답하여 다수의 이전의 디코딩된 스펙트럼 값의 수치 값에 따라 현재 컨텍스트 상태의 계산을 생략하도록 구성된다. 따라서, 컨텍스트 계산은 미리 정해진 조건을 충족하는 다수의 이전의 디코딩된 인접 스펙트럼 값의 그룹의 검출에 응답하여 상당히 간단해진다. 계산량을 줄임으로써, 오디오 신호 디코더의 전력 소비가 또한 감소되어, 모바일 장치에서 상당한 이점을 제공한다.

[0019] 바람직한 실시예에서, 산술 디코더는 현재 컨텍스트 상태를 미리 정해진 조건의 검출을 나타내는 값으로 설정하도록 구성된다. 값의 미리 정해진 범위 내에 있을 수 있는 이와 같은 값으로 컨텍스트 상태를 설정함으로써, 컨텍스트 상태의 나중 평가는 제어될 수 있다. 그러나, 현재 컨텍스트 상태가 설정되는 값은, 값이 미리 정해진 조건의 검출을 나타내는 값의 특징적 범위 내에 있을지라도 또한 다른 기준에 의존할 수 있는 것으로 언급되어야 한다.

[0020] 바람직한 실시예에서, 산술 디코더는 심볼 코드를 디코딩된 스펙트럼 값으로 맵핑하도록 구성된다.

[0021] 바람직한 실시예에서, 산술 디코더는 제 1 시간-주파수 영역의 스펙트럼 값을 평가하여, 개별적으로 또는 종합하여, 크기에 관한 미리 정해진 조건을 충족하는 다수의 스펙트럼 값의 그룹을 검출하도록 구성된다. 산술 디코더는 미리 정해진 조건이 충족되지 않을 경우에 제 1 시간 주파수 영역과 상이한 제 2 시간 주파수 영역의 스펙트럼 값에 따라 컨텍스트 상태를 나타내는 수치 값을 획득하도록 구성된다. 보통 컨텍스트 계산에 이용되는 영역과 상이한 영역 내의 크기에 관한 미리 정해진 조건을 충족하는 다수의 스펙트럼 값의 그룹을 검출하는 것이 바람직한 것으로 발견되었다. 이것은, 비교적 작은 스펙트럼 값, 또는 비교적 큰 스펙트럼 값을 포함하는 영역의 확장, 예컨대, 주파수 확장이 통상적으로 컨텍스트 상태를 나타내는 수치 값의 숫자 계산을 위해 고려될 수 있는 스펙트럼 값의 영역의 치수보다 크다는 사실로 인한 것이다. 따라서, 미리 정해진 조건을 충족하는 다수의 스펙트럼 값의 그룹의 검출, 및 컨텍스트 상태를 나타내는 수치 값의 숫자 계산을 위한 서로 다른 영역을 분석하는 것이 바람직하다(여기서, 숫자 계산은 단지 검출이 비트를 제공하지 않을 경우에 제 2 단계에서 예상될 수 있다).

[0022] 바람직한 실시예에서, 산술 디코더는 컨텍스트 상태에 따라 맵핑 규칙을 선택하기 위해 하나 이상의 해시 테이블(hash tables)을 평가하도록 구성된다. 맵핑 규칙의 선택은 미리 정해진 조건을 충족하는 다수의 인접 스펙트럼 값을 검출하는 메카니즘에 의해 제어될 수 있는 것으로 발견되었다.

[0023] 본 발명에 따른 실시예는 입력 오디오 정보에 기초하여 인코딩된 오디오 정보를 제공하는 오디오 인코더를 생성한다. 오디오 인코더는, 주파수-도메인 오디오 표현이 스펙트럼 값의 세트를 포함하도록 입력 오디오 정보의 시간-도메인 표현에 기초하여 주파수-도메인 오디오 표현을 제공하는 에너지-압축 시간-도메인-대-주파수-도메인 변환기를 포함한다. 오디오 인코더는 또한 가변-길이 코드워드를 이용하여 스펙트럼 값, 또는 이의 사전 처리된 버전을 인코딩하도록 구성되는 산술 인코더를 포함한다. 산술 인코더는 스펙트럼 값 또는 스펙트럼 값의 최상위 비트-플레인(bit-plane)의 값을 코드 값으로 맵핑하도록 구성된다. 산술 인코더는 컨텍스트 상태에 따라 코드 값으로의 스펙트럼 값 또는 스펙트럼 값의 최상위 비트-플레인의 맵핑을 나타내는 맵핑 규칙을 선택하도록 구성된다. 산술 인코더는 다수의 이전의 인코딩된 인접 스펙트럼 값에 따라 현재 컨텍스트 상태를 결정하도록 구성된다. 산술 인코더는, 개별적으로 또는 종합하여, 크기에 관한 미리 정해진 조건을 충족하는 다수의 이전의 인코딩된 인접 스펙트럼 값의 그룹을 검출하여, 검출 결과에 따라 현재 컨텍스트 상태를 결정하도록 구성된다.

[0024] 이러한 오디오 신호 인코더는 상술한 오디오 신호 디코더와 동일한 연구 결과에 기초한다. 오디오 콘텐츠의 디코딩을 위해 효율적인 것으로 나타난 컨텍스트의 적응을 위한 메카니즘은 또한 일관된 시스템(consistent system)을 허용하기 위해 인코더 측에도 적용되어야 하는 것으로 발견되었다.

[0025] 본 발명에 따른 실시예는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하는 방법을 생성한다.

[0026] 본 발명에 따른 다른 실시예는 입력 오디오 정보에 기초하여 인코딩된 오디오 정보를 제공하는 방법을 생성한다.

[0027] 본 발명에 따른 다른 실시예는 상기 방법 중 하나를 수행하기 위한 컴퓨터 프로그램을 생성한다.

발명의 효과

[0028] 방법 및 컴퓨터 프로그램은 상술한 오디오 디코더 및 상술한 오디오 인코더와 동일한 연구 결과에 기초한다.

도면의 간단한 설명

[0029] 본 발명에 따른 실시예들은 이후에 첨부된 도면을 참조로 설명될 것이다.

도 1은 본 발명의 실시예에 따른 오디오 인코더의 개략적인 블록도를 도시한 것이다.

도 2는 본 발명의 실시예에 따른 오디오 디코더의 개략적인 블록도를 도시한 것이다.

도 3은 스펙트럼 값을 디코딩하기 위한 알고리즘 "value_decode()"의 의사-프로그램-코드-표현을 도시한 것이다.

도 4는 상태 계산을 위한 컨텍스트의 개략적 표현을 도시한 것이다.

도 5a는 컨텍스트를 맵핑하기 위한 알고리즘 "arith_map_context()"의 의사-프로그램-코드 표현을 도시한 것이다.

도 5b 및 5c는 컨텍스트 상태 값을 획득하기 위한 알고리즘 "arith_get_context()"의 의사-프로그램-코드 표현을 도시한 것이다.

도 5d는 상태 변수에서 누적-빈도(cumulative-frequency)-테이블 인덱스 값 "pki"을 유도하기 위한 알고리즘 "get_pk(s)"의 의사-프로그램-코드 표현을 도시한 것이다.

도 5e는 상태 변수에서 누적-빈도-테이블 인덱스 값 "pki"을 유도하기 위한 알고리즘 "arith_get_pk(s)"의 의사-프로그램-코드 표현을 도시한 것이다.

도 5f는 상태 변수에서 누적-빈도-테이블 인덱스 값 "pki"을 유도하기 위한 알고리즘 "get_pk(unsigned long s)"의 의사-프로그램-코드 표현을 도시한 것이다.

도 5g는 가변-길이 코드워드에서 심볼을 산술적으로 디코딩하기 위한 알고리즘 "arith_decode ()"의 의사-프로그램-코드 표현을 도시한 것이다.

도 5h는 컨텍스트를 업데이트하기 위한 알고리즘 "arith_update_context()"의 의사-프로그램-코드 표현을 도시한 것이다.

도 5i는 정의 및 변수의 레전드(legend)를 도시한 것이다.

도 6a는 통합된-음성-및-오디오-코딩(USAC) 원시 데이터 블록의 구문 표현을 도시한 것이다.

도 6b는 단일 채널 요소의 구문 표현을 도시한 것이다.

도 6c는 채널 쌍 요소의 구문 표현을 도시한 것이다.

도 6d는 "ics"제어 정보의 구문 표현을 도시한 것이다.

도 6e는 주파수-도메인 채널 스트림의 구문 표현을 도시한 것이다.

도 6f는 산술-코딩된 스펙트럼 데이터의 구문 표현을 도시한 것이다.

도 6g는 스펙트럼 값의 세트를 디코딩하기 위한 구문 표현을 도시한 것이다.

도 6h는 데이터 요소 및 변수의 레전드를 도시한 것이다.

도 7은 본 발명의 다른 실시예에 따른 오디오 인코더의 개략적인 블록도를 도시한 것이다.

도 8은 본 발명의 다른 실시예에 따른 오디오 디코더의 개략적인 블록도를 도시한 것이다.

도 9는 본 발명에 따른 코딩 방식과 USAC 초안 표준(draft standard)의 작업(working) 초안 3에 따른 잡음없는 코딩의 비교를 위한 장치를 도시한 것이다.

도 10a는 USAC 초안 표준의 작업 초안 4에 따라 이용되는 바와 같이 상태 계산을 위한 컨텍스트의 개략적 표현을 도시한 것이다.

도 10b는 본 발명에 따른 실시예에 이용되는 바와 같이 상태 계산을 위한 컨텍스트의 개략적 표현을 도시한 것이다.

도 11a는 USAC 초안 표준의 작업 초안 4에 따른 산술 코딩 기법에 이용되는 바와 같은 테이블의 개요를 도시한 것이다.

도 11b는 본 발명에 따른 산술 코딩 기법에 이용되는 바와 같은 테이블의 개요를 도시한 것이다.

도 12a는 본 발명 및 USAC 초안 표준의 작업 초안 4에 따른 잡음없는 코딩 기법에 대한 읽기 전용 메모리 수요의 그래픽 표현을 도시한 것이다.

도 12b는 본 발명 및 USAC 초안 표준의 작업 초안 4에 의한 개념에 따른 전체 USAC 디코더 데이터 읽기 전용 메모리 수요의 그래픽 표현을 도시한 것이다.

도 13a는 USAC 초안 표준의 작업 초안 3에 따른 산술 코더 및, 본 발명의 실시예에 따른 산술 디코더를 이용하여 통합된-음성-및-오디오-코딩 코더에 의해 이용되는 평균 비트레이트의 테이블 표현을 도시한 것이다.

도 13b는 USAC 초안 표준의 작업 초안 3에 따른 산술 코더 및, 본 발명의 실시예에 따른 산술 코더를 이용하여 통합된-음성-및-오디오-코딩 코더를 위한 비트 저장소(bit reservoir) 제어의 테이블 표현을 도시한 것이다.

도 14는 USAC 초안 표준의 작업 초안 3 및, 본 발명의 실시예에 따른 USAC 코더에 대한 평균 비트레이트의 테이블 표현을 도시한 것이다.

도 15는 프레임 기준으로(on a frame basis) USAC의 최소, 최대 및 평균 비트레이트의 테이블 표현을 도시한 것이다.

도 16은 프레임 기준으로 최상 및 최악의 경우의 테이블 표현을 도시한 것이다.

도 17a 및 17b은 테이블 "ari_s_hash[387]"의 내용의 테이블 표현을 도시한 것이다.

도 18은 테이블 "ari_gs_hash[225]"의 콘텐츠의 테이블 표현을 도시한 것이다.

도 19a 및 19b는 테이블 "ari_cf_m[64][9]"의 콘텐츠의 테이블 표현을 도시한 것이다.

도 20a 및 20b은 테이블 "ari_s_hash[387]의 콘텐츠의 테이블 표현을 도시한 것이다.

발명을 실시하기 위한 구체적인 내용

[0030] 1. 도 7에 따른 오디오 인코더

[0031] 도 7은 본 발명의 실시예에 따른 오디오 인코더의 개략적인 블록도를 도시한 것이다. 오디오 인코더(700)는 입력 오디오 정보(710)를 수신하여, 이에 기초하여, 인코딩된 오디오 정보(712)를 제공하도록 구성된다. 오디오 인코더는, 주파수-도메인 오디오 표현(722)이 스펙트럼 값의 세트를 포함하도록 입력 오디오 정보(710)의 시간-도메인 표현에 기초하여 주파수-도메인 오디오 표현(722)을 제공하도록 구성되는 에너지-압축 시간-도메인-대-주파수-도메인 변환기(720)를 포함한다. 오디오 인코더(700)는 또한 가변-길이 코드워드를 이용하여 (주파수-도메인 오디오 표현(722)을 형성하는 스펙트럼 값의 세트에서의) 스펙트럼 값, 또는 이의 사전 처리된 버전을 인코딩하여, (예컨대, 다수의 가변-길이 코드워드를 포함할 수 있는) 인코딩된 오디오 정보(712)를 획득하도록 구성되는 산술 인코더(730)를 포함한다.

[0032] 산술 인코더(730)는 컨텍스트 상태에 따라 스펙트럼 값 또는 스펙트럼 값의 최상위 비트-플레인의 값을 코드 값(즉, 가변-길이 코드워드)으로 맵핑하도록 구성된다. 산술 인코더(730)는 컨텍스트 상태에 따라 코드 값으

로의 스펙트럼 값 또는 스펙트럼 값의 최상위 비트-플레인의 맵핑을 나타내는 맵핑 규칙을 선택하도록 구성된다. 산술 인코더는 다수의 이전의 인코딩된(바람직하게는, 반드시 필요하지 않지만, 인접한) 스펙트럼 값에 따라 현재 컨텍스트 상태를 결정하도록 구성된다. 이를 위해, 산술 인코더는, 개별적으로 또는 종합하여, 크기에 관한 미리 정해진 조건을 충족하는 다수의 이전의 인코딩된 인접 스펙트럼 값의 그룹을 검출하여, 검출 결과에 따라 현재 컨텍스트 상태를 결정하도록 구성된다.

[0033] 알 수 있는 바와 같이, 코드 값으로의 스펙트럼 값 또는 스펙트럼 값의 최상위 비트-플레인의 맵핑은 맵핑 규칙(742)을 이용하여 스펙트럼 값 인코딩(740)에 의해 수행될 수 있다. 상태 추적기(750)는 컨텍스트 상태를 추적하도록 구성될 수 있고, 개별적으로 또는 종합하여, 크기에 관한 미리 정해진 조건을 충족하는 다수의 이전의 인코딩된 인접 스펙트럼 값의 그룹을 검출하는 그룹 검출기(752)를 포함할 수 있다. 상태 추적기(750)는 또한 바람직하게는 그룹 검출기(752)에 의해 수행되는 상기 검출의 결과에 따라 현재 컨텍스트 상태를 결정하도록 구성된다. 따라서, 상태 추적기(750)는 현재 컨텍스트 상태를 나타내는 정보(754)를 제공한다. 맵핑 규칙 선택기(760)는 코드 값으로의 스펙트럼 값 또는 스펙트럼 값의 최상위 비트-플레인의 맵핑을 나타내는 맵핑 규칙, 예컨대, 누적-빈도-테이블을 선택할 수 있다. 따라서, 맵핑 규칙 선택기(760)는 맵핑 규칙 정보(742)를 스펙트럼 인코딩(740)에 제공한다.

[0034] 상술한 바를 요약하면, 오디오 인코더(700)는 시간-도메인-대-주파수-도메인 변환기에 의해 제공되는 주파수-도메인 오디오 표현의 산술 인코딩을 수행한다. 산술 인코딩은 맵핑 규칙(예컨대, 누적-빈도-테이블)이 이전의 인코딩된 스펙트럼 값에 따라 선택되도록 컨텍스트에 의존적이다. 따라서, 시간 및/또는 주파수에서 (또는 적어도, 미리 정해진 환경 내에서) 서로에 및/또는 현재-인코딩된 스펙트럼 값(즉, 현재 인코딩된 스펙트럼 값의 미리 정해진 환경 내의 스펙트럼 값)에 인접 스펙트럼 값은 산술 인코딩에서 산술 인코딩에 의해 평가되는 확률 분포를 조정하도록 고려된다. 적절한 맵핑 규칙을 선택할 때, 개별적으로 또는 종합하여, 크기에 관한 미리 정해진 조건을 충족하는 다수의 이전의 인코딩된 인접 스펙트럼 값의 그룹이 있는지의 여부를 검출하기 위해 검출이 수행된다. 이러한 검출의 결과는 현재 컨텍스트 상태의 선택, 즉 맵핑 규칙의 선택 시에 적용된다. 특히 작거나 특히 큰 다수의 스펙트럼 값의 그룹이 있는지의 여부를 검출함으로써, 시간-주파수 표현일 수 있는 주파수-도메인 오디오 표현 내의 특정한 특징을 인식할 수 있다. 예컨대, 특히 작거나 특히 큰 다수의 스펙트럼 값의 그룹과 같은 특정한 특징은 이러한 특정 컨텍스트 상태가 특히 양호한 코딩 효율을 제공할 시에 특정 컨텍스트 상태를 이용해야 함을 나타낸다. 따라서, 통상적으로 다수의 이전의 코딩된 스펙트럼 값의 조합에 기초하여 대안적 컨텍스트 평가와 함께 이용되는 미리 정해진 조건을 충족하는 인접 스펙트럼 값의 그룹의 검출은 입력 오디오 정보가 일부 특정한 상태를 취할 경우(예컨대, 큰 마스킹된 주파수 범위를 포함할 경우)에 적절한 컨텍스트의 효율적인 선택을 허용하는 메카니즘을 제공한다.

[0035] 따라서, 효율적인 인코딩은 컨텍스트 계산을 상당히 간단하게 유지하면서 달성될 수 있다.

[0036] 2. 도 8에 따른 오디오 디코더

[0037] 도 8은 오디오 디코더(800)의 개략적인 블록도를 도시한 것이다. 오디오 디코더(800)는 인코딩된 오디오 정보(810)를 수신하여, 이에 기초하여, 디코딩된 오디오 정보(812)를 제공하도록 구성된다. 오디오 디코더(800)는 스펙트럼 값의 산술적-인코딩된 표현(821)에 기초하여 다수의 디코딩된 스펙트럼 값(822)을 제공하도록 구성되는 산술 디코더(820)를 포함한다. 오디오 디코더(800)는 또한 디코딩된 스펙트럼 값(822)을 수신하여, 디코딩된 오디오 정보(812)를 획득하기 위해 디코딩된 스펙트럼 값(822)을 이용하여 디코딩된 오디오 정보를 구성할 수 있는 시간-도메인 오디오 표현(812)을 제공하도록 구성되는 주파수-도메인 대 시간-도메인 변환기(830)를 포함한다.

[0038] 산술 디코더(820)는 스펙트럼 값의 산술적-인코딩된 표현(821)의 코드 값을 디코딩된 스펙트럼 값 중 하나 이상을 나타내는 심볼 코드, 또는 디코딩된 스펙트럼 값 중 하나 이상의 적어도 부분(예컨대, 최상위 비트-플레인)으로 맵핑하도록 구성되는 스펙트럼 값 결정기(determinator)(824)를 포함한다. 스펙트럼 값 결정기(824)는 맵핑 규칙 정보(828a)에 의해 나타낼 수 있는 맵핑 규칙에 따라 맵핑을 수행하도록 구성될 수 있다.

[0039] 산술 디코더(820)는 (컨텍스트 상태 정보(826a)에 의해 나타낼 수 있는) 컨텍스트 상태에 따라 (하나 이상의 스펙트럼 값을 나타내는) 심볼 값으로의 (산술적-인코딩된 표현(821)에 의해 나타내는) 코드-값의 맵핑을 나타내는 맵핑 규칙(예컨대, 누적-빈도-테이블)을 선택하도록 구성된다. 산술 디코더(820)는 다수의 이전의 디코딩된 스펙트럼 값(822)에 따라 현재 컨텍스트 상태를 결정하도록 구성된다. 이를 위해, 상태 추적기(826)가 이용되어, 이전의 디코딩된 스펙트럼 값을 나타내는 정보를 수신한다. 산술 디코더는 또한, 개별적으로 또는 종합하여, 크기에 관한 미리 정해진 조건을 충족하는 다수의 이전의 디코딩된(바람직하게는, 반드시 필요하

지 않지만, 인접한) 스펙트럼 값의 그룹을 검출하여, 검출의 결과에 따라 (예컨대, 컨텍스트 상태 정보(826a)에 의해 나타내는) 현재 컨텍스트 상태를 결정하도록 구성된다.

[0040] 크기에 관한 미리 정해진 조건을 충족하는 다수의 이전의 디코딩된 인접 스펙트럼 값의 그룹의 검출은, 예컨대, 상태 추적기(826)의 부분인 그룹 검출기에 의해 수행될 수 있다. 따라서, 현재 컨텍스트 상태 정보(826a)가 획득된다. 맵핑 규칙의 선택은 맵핑 규칙 선택기(828)에 의해 수행될 수 있으며, 맵핑 규칙 선택기(828)는 현재 컨텍스트 상태 정보(826a)로부터 맵핑 규칙 정보(828a)를 유도하여, 맵핑 규칙 정보(828a)를 스펙트럼 값 결정기(824)에 제공한다.

[0041] 오디오 신호 디코더(800)의 기능에 관하여, 맵핑 규칙이 현재 컨텍스트 상태에 따라 선택되어, 결과적으로 다수의 이전의 디코딩된 스펙트럼 값에 따라 결정됨에 따라, 산술 디코더(820)는, 평균하여, 디코딩되는 스펙트럼 값에 잘 적응되는 맵핑 규칙(예컨대, 누적-빈도-테이블)을 선택하도록 구성된다. 따라서, 디코딩되는 인접 스펙트럼 값 사이의 통계적 의존성(statistical dependencies)이 활용될 수 있다. 더욱이, 개별적으로 또는 종합하여, 크기에 관한 미리 정해진 조건을 충족하는 다수의 이전의 디코딩된 인접 스펙트럼 값의 그룹을 검출함으로써, 이전의 디코딩된 스펙트럼 값의 특정 조건(또는 패턴)에 맵핑 규칙을 적응시킬 수 있다. 예컨대, 다수의 비교적 작은 이전의 디코딩된 인접 스펙트럼 값의 그룹이 식별되거나, 다수의 비교적 큰 이전의 디코딩된 인접 스펙트럼 값의 그룹이 식별될 경우에 특정 맵핑 규칙이 선택될 수 있다. 비교적 큰 스펙트럼 값의 그룹 또는 비교적 작은 스펙트럼 값의 그룹의 존재는 특히 이와 같은 조건에 적응되는 전용 맵핑 규칙이 이용되어야 하는 중요한 표시(significant indication)로 간주될 수 있는 것으로 발견되었다. 따라서, 컨텍스트 계산은 다수의 스펙트럼 값의 이와 같은 그룹의 검출을 활용하여 촉진(또는 가속)될 수 있다. 또한, 오디오 콘텐츠의 특성은 상술한 개념을 적용하지 않고 쉽게 간주될 수 없는 것으로 고려될 수 있다. 예컨대, 개별적으로 또는 종합하여, 크기에 관한 미리 정해진 조건을 충족하는 다수의 스펙트럼 값의 그룹의 검출은 정상 컨텍스트 계산에 이용되는 스펙트럼 값의 세트에 비해 스펙트럼 값의 서로 다른 세트에 기초하여 수행될 수 있다.

[0042] 추가적 상세 사항은 아래에 설명될 것이다.

[0043] 3. 도 1에 따른 오디오 인코더

[0044] 다음에는, 본 발명의 실시예에 따른 오디오 인코더가 설명될 것이다. 도 1은 이와 같은 오디오 인코더(100)의 개략적인 블록도를 도시한 것이다.

[0045] 오디오 인코더(100)는 입력 오디오 정보(110)를 수신하여, 이에 기초하여, 인코딩된 오디오 정보를 구성하는 비트스트림(112)을 제공하도록 구성된다. 오디오 인코더(100)는 선택적으로 입력 오디오 정보(110)를 수신하여, 이에 기초하여, 사전 처리된 입력 오디오 정보(110a)를 제공하도록 구성되는 사전 프로세서(preprocessor)(120)를 포함한다. 오디오 인코더(100)는 또한 신호 변환기로서 명시되는 에너지-압축 시간-도메인 대 주파수-도메인 신호 변환기(130)를 포함한다. 신호 변환기(130)는 입력 오디오 정보(110, 110a)를 수신하여, 이에 기초하여, 바람직하게는 스펙트럼 값의 세트의 형식을 취하는 주파수-도메인 오디오 정보(132)를 제공하도록 구성된다. 예컨대, 신호 변환기(130)는 입력 오디오 정보(110, 110a)의 프레임(예컨대, 시간-도메인 샘플의 블록)을 수신하여, 각각의 오디오 프레임의 오디오 콘텐츠를 나타내는 스펙트럼 값의 세트를 제공하도록 구성될 수 있다. 게다가, 신호 변환기(130)는 입력 오디오 정보(110, 110a)의 다수의 다음 중복 또는 비중복 오디오 프레임을 수신하여, 이에 기초하여, 스펙트럼 값의 다음 세트의 시퀀스를 포함하는 시간-주파수-도메인 오디오 표현을 제공하도록 구성될 수 있으며, 스펙트럼 값의 한 세트는 각 프레임과 관련된다.

[0046] 에너지-압축 시간-도메인 대 주파수-도메인 신호 변환기(130)는 서로 다른 중복 또는 비중복 주파수 범위와 관련된 스펙트럼 값을 제공하는 에너지-압축 필터뱅크를 포함할 수 있다. 예컨대, 신호 변환기(130)는 변환 윈도우를 이용하여 입력 오디오 정보(110, 110a)(또는 이의 프레임)를 윈도잉하여, 윈도잉된 입력 오디오 정보(110, 110a)(또는 이의 윈도잉된 프레임)의 수정된-이산-코사인-변환을 수행하도록 구성되는 윈도잉 MDCT 변환기(130a)를 포함할 수 있다. 따라서, 주파수-도메인 오디오 표현(132)은 입력 오디오 정보의 프레임과 관련된 MDCT 계수의 형식의 예컨대 1024 스펙트럼 값의 세트를 포함할 수 있다.

[0047] 오디오 인코더(100)는, 선택적으로, 주파수-도메인 오디오 표현(132)을 수신하여, 이에 기초하여, 사후 처리된 주파수-도메인 오디오 표현(142)을 제공하도록 구성되는 스펙트럼 사후 프로세서(post-processor)(140)를 더 포함할 수 있다. 스펙트럼 사후 프로세서(140)는, 예컨대, 본 기술 분야에서 알려져 있는 시간적 잡음 형상화(shaping) 및/또는 장기(long term) 예측 및/또는 어떤 다른 스펙트럼 사후 처리를 수행하도록 구성될 수 있다. 오디오 인코더는, 선택적으로, 주파수-도메인 오디오 표현(132) 또는 이의 사후 처리된 버전(142)을 수

신하여, 스케일링 및 양자화된 주파수-도메인 오디오 표현(152)을 제공하도록 구성되는 스케일러/양자화기(150)를 더 포함한다.

[0048] 오디오 인코더(100)는, 선택적으로, 입력 오디오 정보(110)(또는 이의 사후 처리된 버전(110a))를 수신하여, 이에 기초하여, 선택적 제어 정보를 제공하도록 구성되는 음향 심리학(psycho-acoustic) 모델 프로세서(160)를 더 포함하며, 선택적 제어 정보는 에너지-압축 시간-도메인 대 주파수-도메인 신호 변환기(130)의 제어, 선택적 스펙트럼 사후 프로세서(140)의 제어 및/또는 선택적 스케일러/양자화기(150)의 제어를 위해 이용될 수 있다. 예컨대, 음향 심리학 모델 프로세서(160)는 입력 오디오 정보를 분석하여, 입력 오디오 정보(110, 110a)의 어느 구성 요소가 특히 오디오 콘텐츠의 인간의 인지(human perception)에 중요한지를 판정하고, 입력 오디오 정보(110, 110a)의 어느 구성 요소가 오디오 콘텐츠의 인지에 덜 중요한지를 판정하도록 구성될 수 있다. 따라서, 음향 심리학 모델 프로세서(160)는, 스케일러/양자화기(150)에 의한 주파수-도메인 오디오 표현(132, 142)의 스케일링 및/또는 스케일러/양자화기(150)에 의해 적용되는 양자화 해상도를 조정하기 위해 오디오 인코더(100)에 의해 이용되는 제어 정보를 제공할 수 있다. 결과적으로, 인지적으로 중요한 스케일 팩터 대역(즉, 특히 오디오 콘텐츠의 인간의 인지에 중요한 인접 스펙트럼 값의 그룹)은 큰 스케일링 팩터로 스케일링되고, 비교적 높은 해상도로 양자화되지만, 인지적으로 덜 중요한 스케일 팩터 대역(즉, 인접 스펙트럼 값의 그룹)은 비교적 작은 스케일링 팩터로 스케일링되고, 비교적 낮은 해상도로 양자화된다. 따라서, 인지적으로 더 중요한 주파수의 스케일링된 스펙트럼 값은 통상적으로 인지적으로 덜 중요한 주파수의 스펙트럼 값보다 상당히 더 크다.

[0049] 오디오 인코더는 또한, 주파수-도메인 오디오 표현(132)의 스케일링 및 양자화된 버전(152) (또는, 대안적으로, 주파수-도메인 오디오 표현(132)의 사후 처리된 버전(142), 또는 심지어 주파수-도메인 오디오 표현(132) 자체)을 수신하여, 이에 기초하여 산술 코드워드 정보(172a)를 제공함으로써, 산술 코드워드 정보가 주파수-도메인 오디오 표현(152)을 나타내도록 구성되는 산술 인코더(170)를 포함한다.

[0050] 오디오 인코더(100)는 또한 산술 코드워드 정보(172a)를 수신하도록 구성되는 비트스트림 페이로드 포맷터(190)를 포함한다. 비트스트림 페이로드 포맷터(190)는 또한 통상적으로, 예컨대, 스케일 팩터가 스케일러/양자화기(150)에 의해 적용되었음을 나타내는 스케일 팩터 정보와 같은 추가적인 정보를 수신하도록 구성된다. 게다가, 비트스트림 페이로드 포맷터(190)는 다른 제어 정보를 수신하도록 구성될 수 있다. 비트스트림 페이로드 포맷터(190)는 원하는 비트스트림 구문에 따라 비트스트림을 어셈블링함으로써 수신된 정보에 기초하여 비트스트림(112)을 제공하도록 구성되며, 이에 대해서는 아래에서 논의될 것이다.

[0051] 다음에는, 산술 인코더(170)에 관한 상세 사항이 설명될 것이다. 산술 인코더(170)는 주파수-도메인 오디오 표현(132)의 다수의 사후 처리 및 스케일링 및 양자화된 스펙트럼 값을 수신하도록 구성된다. 산술 인코더는 스펙트럼 값에서 최상위 비트-플레인 m을 추출하도록 구성되는 최상위-비트-플레인-추출기(174)를 포함한다. 여기서, 최상위 비트-플레인은 스펙트럼 값의 최상위 비트인 하나 이상의 비트(예컨대, 2 또는 3 비트)를 포함할 수 있는 것으로 언급되어야 한다. 따라서, 최상위-비트-플레인-추출기(174)는 스펙트럼 값의 최상위 비트-플레인 값(176)을 제공한다.

[0052] 산술 인코더(170)는 또한 최상위 비트-플레인 값 m을 나타내는 산술 코드워드 acod_m [pki][m]를 결정하도록 구성되는 제 1 코드워드 결정기(180)를 포함한다. 선택적으로, 코드워드 결정기(180)는 또한, 예컨대, 많은 하위 비트-플레인이 이용 가능한 방법을 나타내는 (결과적으로, 최상위 비트-플레인의 숫자 가중(numeric weight)을 나타내는) 하나 이상의 이스케이프(escape) 코드워드(또한 여기서 "ARITH_ESCAPE"로 명시됨)를 제공할 수 있다. 제 1 코드워드 결정기(180)는 누적-빈도-테이블 인덱스 pki를 가진 (또는 이에 의해 참조되는) 선택된 누적-빈도-테이블을 이용하여 최상위 비트-플레인 값과 관련된 코드워드를 제공하도록 구성될 수 있다.

[0053] 어떤 누적-빈도-테이블이 선택되어야 하는지에 관해 결정하기 위해, 산술 인코더는 바람직하게는, 예컨대, 어떤 스펙트럼 값이 이전에 인코딩되었는지를 관찰하여 산술 인코더의 상태를 추적하도록 구성되는 상태 추적기(182)를 포함한다. 상태 추적기(182)는 결과적으로 상태 정보(184), 예컨대, "s" 또는 "t"로 명시되는 상태 값을 제공한다. 산술 인코더(170)는 또한 상태 정보(184)를 수신하여, 선택된 누적-빈도-테이블을 나타내는 정보(188)를 코드워드 결정기(180)에 제공하도록 구성되는 누적-빈도-테이블 선택기(186)를 포함한다. 예컨대, 누적-빈도-테이블 선택기(186)는 64 누적-빈도-테이블의 세트에서 어떤 누적-빈도-테이블이 코드워드 결정기에 의해 사용하기 위해 선택되는지를 나타내는 누적-빈도-테이블 인덱스 "pki"를 제공할 수 있다. 대안적으로, 누적-빈도-테이블 선택기(186)는 전체 선택된 누적-빈도-테이블을 코드워드 결정기에 제공할 수 있다. 따라서, 코드워드 결정기(180)는 최상위 비트-플레인 값 m의 코드워드 acod_m[pki][m]의 제공을 위해

선택된 누적-빈도-테이블을 이용하여, 최상위 비트-플레인 값 m을 인코딩하는 실제 코드워드 acod_m[pki][m]가 m의 값 및 누적-빈도-테이블 인덱스 pki에 의존하여, 결과적으로 현재 상태 정보(184)에 의존하도록 할 수 있다. 코딩 프로세스 및 획득된 코드워드 포맷에 관한 추가적 상세 사항이 아래에 설명될 것이다.

[0054] 인코딩되는 스펙트럼 값 중 하나 이상이 최상위 비트-플레인만을 이용하여 인코딩 가능한 값의 범위를 초과할 경우에, 산술 인코더(170)는 스케일링 및 양자화된 주파수-도메인 오디오 표현(152)에서 하나 이상의 하위 비트-플레인을 추출하도록 구성되는 하위 비트-플레인 추출기(189a)를 더 포함한다. 하위 비트-플레인은 원하는 대로 하나 이상의 비트를 포함할 수 있다. 따라서, 하위 비트-플레인 추출기(189a)는 하위 비트-플레인 정보(189b)를 제공한다. 산술 인코더(170)는 또한, 하위 비트-플레인 정보(189d)를 수신하여, 이에 기초하여, 0, 1 이상의 하위 비트-플레인의 콘텐츠를 나타내는 0, 1 이상의 코드워드 "acod_r"를 제공하도록 구성되는 제 2 코드워드 결정기(189c)를 포함한다. 제 2 코드워드 결정기(189c)는 하위 비트-플레인 정보(189b)로부터 하위 비트-플레인 코드워드 "acod_r"를 유도하기 위해 산출 인코딩 알고리즘 또는 어떤 다른 인코딩 알고리즘을 적용하도록 구성될 수 있다.

[0055] 여기서, 인코딩되는 스케일링 및 양자화된 스펙트럼 값이 비교적 작을 경우에는 하위 비트-플레인이 전혀 없을 수 있고, 인코딩되는 스케일링 및 양자화된 스펙트럼 값이 중간 범위일 경우에는 하나의 하위 비트-플레인이 있을 수 있으며, 인코딩되는 스케일링 및 양자화된 스펙트럼 값이 비교적 큰 값을 취할 경우에는 하나 이상의 하위 비트-플레인이 있을 수 있도록 하위 비트-플레인 코드워드의 수는 스케일링 및 양자화된 스펙트럼 값(152)의 값에 따라 변할 수 있는 것으로 언급되어야 한다.

[0056] 상술한 바를 요약하면, 산술 인코더(170)는 계층적(hierarchical) 인코딩 프로세스를 이용하여 정보(152)로 나타내는 스케일링 및 양자화된 스펙트럼 값을 인코딩하도록 구성된다. (예컨대, 스펙트럼 값마다 1, 2 또는 3 비트를 포함하는) 최상위 비트-플레인은 최상위 비트-플레인 값의 산술 코드워드 "acod_m[pki][m]"를 획득하도록 인코딩된다. 하나 이상의 하위 비트-플레인(하위 비트-플레인의 각각은, 예컨대, 1, 2 또는 3 비트를 포함한다)은 하나 이상의 코드워드 "acod_r"를 획득하도록 인코딩된다. 최상위 비트-플레인을 인코딩하면, 최상위 비트-플레인의 값 m은 코드워드 acod_m[pki][m]로 맵핑된다. 이를 위해, 64의 서로 다른 누적-빈도-테이블은 산술 인코더(170)의 상태, 즉 이전의 인코딩된 스펙트럼 값에 따라 값 m의 인코딩을 위해 이용가능하다. 따라서, 코드워드 "acod_m[pki][m]"가 획득된다. 게다가, 하나 이상의 코드워드 "acod_r"가 제공되어, 하나 이상의 하위 비트-플레인이 제공될 경우에 비트스트림 내에 포함된다.

[0057] 재설정 설명

[0058] 오디오 인코더(100)는 선택적으로, 컨텍스트를 재설정함으로써, 예컨대, 상태 인덱스를 디폴트 값으로 설정함으로써 비트레이트의 개선이 획득될 수 있는지의 여부를 판단하도록 구성될 수 있다. 따라서, 오디오 인코더(100)는 산술 인코딩을 위한 컨텍스트가 재설정되는지의 여부를 나타내고, 또한 상응하는 디코더에서 산술 디코딩을 위한 컨텍스트가 재설정되어야 하는지의 여부를 나타내는 재설정 정보(예컨대, "arith_reset_flag"라 명명함)를 제공하도록 구성될 수 있다.

[0059] 비트스트림 포맷 및 적용된 누적-빈도 테이블에 관한 상세 사항은 아래에서 논의될 것이다.

[0060] 4. 오디오 디코더

[0061] 다음에는, 본 발명의 실시예에 따른 오디오 디코더가 설명된다. 도 2는 이와 같은 오디오 디코더(200)의 개략적 블록도를 도시한다.

[0062] 오디오 디코더(200)는 인코딩된 오디오 정보를 나타내며, 오디오 인코더(100)에 의해 제공되는 비트스트림(112)과 동일할 수 있는 비트스트림(210)을 수신하도록 구성된다. 오디오 디코더(200)는 비트스트림(210)에 기초하여 디코딩된 오디오 정보(212)를 제공한다.

[0063] 오디오 디코더(200)는 비트스트림(210)을 수신하여, 비트스트림(210)에서 인코딩된 주파수-도메인 오디오 표현(222)을 추출하도록 구성되는 선택적인 비트스트림 페이로드 디포맷터(220)를 포함한다. 예컨대, 비트스트림 페이로드 디포맷터(220)는, 예컨대, 스펙트럼 값 a의 최상위 비트-플레인 값 m을 나타내는 산술 코드워드 "acod_m [pki][m]", 및 주파수-도메인 오디오 표현의 스펙트럼 값 a의 하위 비트-플레인의 콘텐츠를 나타내는 코드 워드 "acod_r"와 같은 산술적-코딩된 스펙트럼 값을 비트스트림(210)에서 추출하도록 구성될 수 있다. 따라서, 인코딩된 주파수-도메인 오디오 표현(222)은 스펙트럼 값의 산술적-인코딩된 표현을 구성(또는 포함)한다. 비트스트림 페이로드 디포맷터(220)는 도 2에 도시되지 않은 추가적 제어 정보를 비트스트림에서 추출하도록 더 구성된다. 게다가, 비트스트림 페이로드 디포맷터는 선택적으로 또한 산술 재설정 플래그 또는

"arith_reset_flag"으로 명시되는 상태 재설정 정보(224)를 비트스트림(210)에서 추출하도록 구성된다.

[0064] 오디오 디코더(200)는 또한 "스펙트럼 잡음없는 디코더"로 명시되는 산술 디코더(230)를 포함한다. 산술 디코더(230)는 인코딩된 주파수-도메인 오디오 표현(220) 및, 선택적으로, 상태 재설정 정보(224)를 수신하도록 구성된다. 산술 디코더(230)는 또한 스펙트럼 값의 디코딩된 표현을 포함할 수 있는 디코딩된 주파수-도메인 오디오 표현(232)을 제공하도록 구성된다. 예컨대, 디코딩된 주파수-도메인 오디오 표현(232)은 인코딩된 주파수-도메인 오디오 표현(220)으로 나타내는 스펙트럼 값의 디코딩된 표현을 포함할 수 있다.

[0065] 오디오 디코더(200)는 또한 디코딩된 주파수-도메인 오디오 표현(232)을 수신하여, 이에 기초하여, 역 양자화 및 리스케일링된(rescaled) 주파수-도메인 오디오 표현(242)을 제공하도록 구성되는 선택적 역 양자화기/리스케일러(240)를 포함한다.

[0066] 오디오 디코더(200)는 역 양자화 및 리스케일링된 주파수-도메인 오디오 표현(242)을 수신하여, 이에 기초하여, 역 양자화 및 리스케일링된 주파수-도메인 오디오 표현(242)의 사전 처리된 버전(252)을 제공하도록 구성되는 선택적 스펙트럼 사전 프로세서(250)를 더 포함한다. 오디오 디코더(200)는 또한 "신호 변환기"로서 명시되는 주파수-도메인 대 시간-도메인 신호 변환기(260)를 포함한다. 신호 변환기(260)는 역 양자화 및 리스케일링된 주파수-도메인 오디오 표현(242)의 사전 처리된 버전(252)(또는, 대안적으로, 역 양자화 및 리스케일링된 주파수-도메인 오디오 표현(242) 또는 디코딩된 주파수-도메인 오디오 표현(232))을 수신하여, 이에 기초하여, 오디오 정보의 시간-도메인 표현(262)을 제공하도록 구성된다. 주파수-도메인 대 시간-도메인 신호 변환기(260)는, 예컨대, (예컨대, 중복-및-추가와 같은 다른 보조 기능뿐만 아니라) 역-수정된-이산-코사인 변환(IMDCT) 및 적절한 윈도잉을 수행하는 변환기를 포함할 수 있다.

[0067] 오디오 디코더(200)는 오디오 정보의 시간-도메인 표현(262)을 수신하여, 시간-도메인 사후 처리를 이용하여 디코딩된 오디오 정보(212)를 획득하도록 구성되는 선택적 시간-도메인 사후 프로세서(270)를 더 포함할 수 있다. 그러나, 사후 처리가 생략되면, 시간-도메인 표현(262)은 디코딩된 오디오 정보(212)와 동일할 수 있다.

[0068] 역 양자화기/리스케일러(240), 스펙트럼 사전 프로세서(250), 주파수-도메인 대 시간-도메인 신호 변환기(260) 및 시간-도메인 사후 프로세서(270)는 비트스트림 페이로드 디포맷터(220)에 의해 비트스트림(210)에서 추출되는 제어 정보에 따라 제어될 수 있다.

[0069] 오디오 디코더(200)의 전체 기능을 요약하면, 디코딩된 주파수-도메인 오디오 표현(232), 예컨대, 인코딩된 오디오 정보의 오디오 프레임과 관련된 스펙트럼 값의 세트는 산술 디코더(230)를 이용하여 인코딩된 주파수-도메인 표현(222)에 기초하여 획득될 수 있다. 그 다음에, 예컨대, MDCT 계수일 수 있는 1024 스펙트럼 값의 세트는 역 양자화, 리스케일링 및 사전 처리된다. 따라서, 스펙트럼 값(예컨대, 1024 MDCT 계수)의 역 양자화, 리스케일링 및 스펙트럼 사전 처리된 세트가 획득된다. 그 후에, 오디오 프레임의 시간-도메인 표현은 주파수-도메인 값(예컨대, MDCT 계수)의 역 양자화, 리스케일링 및 스펙트럼 사전 처리된 세트로부터 유도된다. 따라서, 오디오 프레임의 시간-도메인 표현이 획득된다. 주어진 오디오 프레임의 시간-도메인 표현은 이전 및/또는 다음 오디오 프레임의 시간-도메인 표현과 조합될 수 있다. 예컨대, 다음 오디오 프레임의 시간-도메인 표현 사이의 중복-및-추가는 인접한 오디오 프레임의 시간-도메인 표현 사이의 전환을 평활화(smoothen)하고, 앨리어싱 소거를 획득하기 위해 수행될 수 있다. 디코딩된 시간-주파수 도메인 오디오 표현(232)에 기초하여 디코딩된 오디오 정보(212)의 재구성에 관한 상세 사항에 대해, 예컨대, International Standard ISO/IEC 14496-3, part 3, sub-part 4에 대한 참조가 행해지며, 여기서, 상세한 논의가 주어진다. 그러나, 다른 더욱 정교한 중복 및 앨리어싱 -소거 기법이 이용될 수 있다.

[0070] 다음에는, 산술 디코더(230)에 관한 몇몇 상세 사항이 설명될 것이다. 산술 디코더(230)는 최상위 비트-플레인 값 m을 나타내는 산술 코드워드 acod_m [pki][m]을 수신하도록 구성되는 최상위 비트-플레인 결정기(284)를 포함한다. 최상위 비트-플레인 결정기(284)는 산술 코드워드 "acod_m [pki][m]"에서 최상위 비트-플레인 값 m을 유도하기 위해 다수의 64 누적-빈도-테이블을 포함하는 세트에서 누적-빈도-테이블을 이용하도록 구성될 수 있다.

[0071] 최상위 비트-플레인 결정기(284)는 코드워드 acod_m에 기초하여 스펙트럼 값의 최상위 비트-플레인의 값(286)을 유도하도록 구성된다. 산술 디코더(230)는 스펙트럼 값의 하나 이상의 하위 비트-플레인을 나타내는 하나 이상의 코드워드 "acod_r"를 수신하도록 구성되는 하위 비트-플레인 결정기(288)를 더 포함한다. 따라서, 하위 비트-플레인 결정기(288)는 하나 이상의 하위 비트-플레인의 디코딩된 값(290)을 제공하도록 구성된다. 오디오 디코더(200)는 또한, 스펙트럼 값의 최상위 비트-플레인의 디코딩된 값(286) 및, 이와 같은 하위 비트

-플레인이 현재 스펙트럼 값에 이용 가능할 경우에 스펙트럼 값의 하나 이상의 하위 비트-플레인의 디코딩된 값(290)을 수신하도록 구성되는 비트-플레인 조합기(292)를 포함한다. 따라서, 비트-플레인 조합기(292)는 디코딩된 주파수-도메인 오디오 표현(232)의 부분인 디코딩된 스펙트럼 값을 제공한다. 물론, 산술 디코더(230)는 통상적으로 오디오 콘텐츠의 현재 프레임과 관련된 디코딩된 스펙트럼 값의 전체 세트를 획득하기 위해 다수의 스펙트럼 값을 제공하도록 구성된다.

[0072] 산술 디코더(230)는 산술 디코더의 상태를 나타내는 상태 인덱스(298)에 따라 64 누적-빈도 테이블 중 하나를 선택하도록 구성되는 누적-빈도-테이블 선택기(296)를 더 포함한다. 산술 디코더(230)는 이전의 디코딩된 스펙트럼 값에 따라 산술 디코더의 상태를 추적하도록 구성되는 상태 추적기(299)를 더 포함한다. 상태 정보는 선택적으로 상태 재설정 정보(224)에 응답하여 디폴트 상태 정보로 재설정될 수 있다. 따라서, 누적-빈도-테이블 선택기(296)는 코드워드 "acod_m"에 따라 최상위 비트-플레인 값 m의 디코딩 시에 적용하기 위해 선택된 누적-빈도-테이블의 인덱스(예컨대,pki) 또는 선택된 누적-빈도-테이블 자체를 제공하도록 구성된다.

[0073] 오디오 디코더(200)의 기능을 요약하면, 오디오 디코더(200)는 비트레이트-효율적-인코딩된 주파수-도메인 오디오 표현(222)을 수신하여, 이에 기초하여 디코딩된 주파수-도메인 오디오 표현을 획득하도록 구성된다. 인코딩된 주파수-도메인 오디오 표현(222)에 기초하여 디코딩된 주파수-도메인 오디오 표현(232)을 획득하기 위해 이용되는 산술 디코더(230)에서, 인접 스펙트럼 값의 최상위 비트-플레인의 값의 서로 다른 조합의 확률(probability)은 누적-빈도-테이블을 적용하도록 구성되는 산술 디코더(280)를 이용하여 활용된다. 환언하면, 스펙트럼 값 사이에 통계적 의존성(statistic dependencies)은 이전에 계산된 디코딩된 스펙트럼 값을 관찰하여 획득되는 상태 인덱스(298)에 따라 64개의 서로 다른 누적-빈도-테이블로 구성된 세트에서 서로 다른 누적-빈도-테이블을 선택하여 활용된다.

[0074] 5. 스펙트럼 잡음없는 코딩의 툴에 관한 개요

[0075] 다음에는, 예컨대, 산술 인코더(170) 및 산술 디코더(230)에 의해 수행되는 인코딩 및 디코딩 알고리즘에 관한 상세 사항이 설명될 것이다.

[0076] 디코딩 알고리즘의 설명에 집중한다. 그러나, 상응하는 인코딩 알고리즘은 디코딩 알고리즘의 요지에 따라 수행될 수 있으며, 여기서, 맵핑은 역으로 되는 것으로 언급되어야 한다.

[0077] 다음에 논의되는 디코딩은 통상적으로 사후 처리 및 스케일링 및 양자화된 스펙트럼 값의 소위 "스펙트럼 잡음없는 코딩"을 허용하기 위해 이용되는 것으로 언급되어야 한다. 스펙트럼 잡음없는 코딩은, 예컨대, 에너지-압축 시간-도메인 대 주파수-도메인 신호 변환기에 의해 획득되는 양자화된 스펙트럼의 중복을 더 줄이기 위해 오디오 인코딩/디코딩 개념에 이용된다.

[0078] 본 발명의 실시예에 이용되는 스펙트럼 잡음없는 코딩 기법은 동적-적응된 컨텍스트와 함께 산술 코딩에 기초한다. 잡음없는 코딩은 양자화된 스펙트럼 값(의 원래 또는 인코딩된 표현)에 의해 공급되고, 예컨대, 다수의 이전의 디코딩된 이웃한 스펙트럼 값에서 유도된 컨텍스트-의존 누적-빈도-테이블을 이용한다. 여기서, 시간 및 주파수 양방에서의 이웃(neighborhood)은 도 4에 예시된 바와 같이 고려된다. 그 후, (아래에 설명되는) 누적-빈도-테이블은, 산술 코더에 의해 가변-길이 이진 코드를 생성하고, 산술 디코더에 의해 가변-길이 이진 코드에서 디코딩된 값을 유도하는데 이용된다.

[0079] 예컨대, 산술 코더(170)는 각각의 확률에 따라 심볼의 주어진 세트에 대한 이진 코드를 생성한다. 이진 코드는 심볼의 세트가 있는 확률 구간을 코드 워드에 맵핑하여 생성된다.

[0080] 다음에는, 스펙트럼 잡음없는 코딩의 툴의 다른 짧은 개요가 주어질 것이다. 스펙트럼 잡음없는 코딩은 양자화 스펙트럼의 중복성을 추가로 줄이기 위해 이용된다. 스펙트럼 잡음없는 코딩 기법은 동적으로 적응된 컨텍스트와 함께 산술 코딩에 기초한다. 잡음없는 코딩은 양자화된 스펙트럼 값에 의해 공급되고, 예컨대, 7의 이전의 디코딩된 이웃한 스펙트럼 값에서 유도된 컨텍스트 의존 누적-빈도-테이블을 이용한다.

[0081] 여기서, 시간 및 주파수 양방에서의 이웃은 도 4에 예시된 바와 같이 고려된다. 그 후, 누적-빈도-테이블은 산술 코더에 의해 가변 길이 이진 코드를 생성하는데 이용된다.

[0082] 산술 코더는 심볼의 주어진 세트 및 이들의 각각의 확률에 대한 이진 코드를 생성한다. 이진 코드는 심볼의 세트가 있는 확률 구간을 코드 워드에 맵핑하여 생성된다.

[0083] 6. 디코딩 프로세스

[0084] 6.1 디코딩 프로세스 개요

[0085] 다음에는, 스펙트럼 값을 디코딩하는 프로세스의 개요가 도 3을 참조로 주어지며, 도 3은 다수의 스펙트럼 값을 디코딩하는 프로세스의 의사-프로그램 코드 표현을 도시한다.

[0086] 다수의 스펙트럼 값을 디코딩하는 프로세스는 컨텍스트의 초기화(310)를 포함한다. 컨텍스트의 초기화(310)는 함수 "arith_map_context(lg)"를 이용하여 이전의 컨텍스트에서의 현재 컨텍스트의 유도를 포함한다. 이전의 컨텍스트에서의 현재 컨텍스트의 유도는 컨텍스트의 재설정을 포함할 수 있다. 컨텍스트의 재설정 및 이전의 컨텍스트에서의 현재 컨텍스트의 유도의 양방에 대해서는 아래에서 논의될 것이다.

[0087] 다수의 스펙트럼 값의 디코딩은 또한 스펙트럼 값 디코딩(312) 및 컨텍스트 업데이트(314)의 반복을 포함하며, 컨텍스트 업데이트는 아래에 설명되는 함수 "Arith_update_context(a,i,lg)"에 의해 수행된다. 스펙트럼 값 디코딩(312) 및 컨텍스트 업데이트(314)는 lg 번 반복되며, lg는 (예컨대, 오디오 프레임에 대해) 디코딩되는 스펙트럼 값의 수를 나타낸다. 스펙트럼 값 디코딩(312)은 컨텍스트-값 계산(312a), 최상위 비트-플레인 디코딩(312b), 및 하위 비트-플레인 추가(312c)를 포함한다.

[0088] 상태 값 계산(312a)은 함수 "arith_get_context(i, lg, arith_reset_flag, N/2)"를 이용하는 제 1 상태 값 s의 계산을 포함하며, 이러한 함수는 제 1 상태 값 s을 복귀(return)시킨다. 상태 값 계산(312a)은 또한 레벨 값 "lev0" 및 레벨 값 "lev"의 계산을 포함하며, 레벨 값 "lev0", "lev"은 제 1 상태 값 s을 24 비트만큼 오른쪽으로 시프트하여 획득된다. 상태 값 계산(312a)은 또한 참조 번호(312a)에서 도 3에 도시된 식에 따라 제 2 상태 값 t의 계산을 포함한다.

[0089] 최상위 비트-플레인 디코딩(312b)은 변수 j가 알고리즘(312ba)의 제 1 실행 전에 0으로 초기화되는 디코딩 알고리즘(312ba)의 반복 실행을 포함한다.

[0090] 알고리즘(312ba)은, 아래에서 논의되는 함수 "arith_get_pk()"를 이용하여 제 2 상태 값 t 및 레벨 값 "lev" 및 lev0에 따라 (또한 누적-빈도-테이블 인덱스 역할을 하는) 상태 인덱스 "pki"의 계산을 포함한다. 알고리즘(312ba)은 또한 상태 인덱스 pki에 따라 누적-빈도-테이블의 선택을 포함하며, 여기서, 변수 "cum_freq"는 상태 인덱스 pki에 따라 64 누적-빈도-테이블 중 하나의 시작 어드레스로 설정될 수 있다. 또한, 변수 "cfl"는, 예컨대, 알파벳의 심볼의 수, 즉 디코딩될 수 있는 서로 다른 값의 수와 동일한 선택된 누적-빈도-테이블의 길이로 초기화될 수 있다. 최상위 비트-플레인 값 m의 디코딩을 위해 이용 가능한, "arith_cf_m[pki=0][9]"에서 "arith_cf_m[pki=63][9]"까지의 모든 누적-빈도-테이블의 길이는 8 개의 서로 다른 최상위 비트-플레인 값 및 이스케이프 심볼이 디코딩될 수 있음에 따라 9이다. 그 다음, 최상위 비트-플레인 값 m은 함수 "arith_decode()"를 실행하고, (변수 "cum_freq" 및 변수 "cfl"로 나타내는) 선택된 누적-빈도-테이블을 고려하여 획득될 수 있다. 최상위 비트-플레인 값 m을 유도하면, 비트스트림(210)의 "acod_m"이라는 비트가 평가될 수 있다(예컨대, 도 6g를 참조).

[0091] 알고리즘(312ba)은 또한 최상위 비트-플레인 값 m이 이스케이프 심볼 "ARITH_ESCAPE"과 동일한지 동일하지 않은지를 검사하는 것을 포함한다. 최상위 비트-플레인 값 m이 산술 이스케이프 심볼과 동일하지 않으면, 알고리즘(312ba)은 중단되어("break"-condition), 알고리즘(312ba)의 나머지 명령어는 스킵(skip)된다. 따라서, 프로세스의 실행은 최상위 비트-플레인 값 m과 동일할 스펙트럼 값 a(명령어 "a=m")의 설정으로 계속된다. 대조적으로, 디코딩된 최상위 비트-플레인 값 m이 산술 이스케이프 심볼 "ARITH_ESCAPE"과 동일하면, 레벨 값 "lev"은 1씩 증가된다. 상술한 바와 같이, 그 후, 알고리즘(312ba)은 디코딩된 최상위 비트-플레인 값 m이 산술 이스케이프 심볼과 상이할 때까지 반복된다.

[0092] 최상위 비트-플레인 디코딩이 완료되자마자, 즉, 산술 이스케이프 심볼과 상이한 최상위 비트-플레인 값 m이 디코딩되자마자, 스펙트럼 값 변수 "a"는 최상위 비트-플레인 값 m과 동일하도록 설정된다. 그 다음, 예컨대, 도 3에서 참조 번호(312c)에 도시된 바와 같이 하위 비트-플레인이 획득된다. 스펙트럼 값의 각 하위 비트-플레인의 경우, 두 이진 값 중 하나는 디코딩된다. 예컨대, 하위 비트-플레인 값 r이 획득된다. 그 다음, 스펙트럼 값 변수 "a"는 스펙트럼 값 변수 "a"의 콘텐츠를 1 비트씩 왼쪽으로 시프트하고, 현재-디코딩된 하위 비트-플레인 값 r을 최하위 비트로서 추가하여 업데이트된다. 그러나, 하위 비트-플레인의 값을 획득하기 위한 개념은 본 발명에 대한 특정 관련성이 없다는 것으로 언급되어야 한다. 일부 실시예에서, 어떤 하위 비트-플레인의 디코딩은 심지어 생략될 수 있다. 대안적으로, 이를 위해 서로 다른 디코딩 알고리즘이 이용될 수 있다.

[0093] 6.2 도 4에 따른 디코딩 순서

[0094] 다음에는, 스펙트럼 값의 디코딩 순서가 설명될 것이다.

[0095] 스펙트럼 계수는 잡음없이 코딩되어, 최저 주파수 계수로부터 시작하여 최고 주파수 계수로 진행하여 (예컨대, 비트스트림으로) 전송된다.

[0096] (예컨대, ISO/IEC 14496, part3, subpart 4에서 논의된 바와 같이 수정된-이산-코사인-변환을 이용하여 획득되는) 고급 오디오 코딩에서의 계수는 "x_ac_quant[g][win][sfb][bin]"이라는 어레이에 저장되고, 잡음없는-코딩-코드워드 (예컨대, acod_m, acod_r)의 전송 순서는 이들이 어레이에 수신되고 저장된 순서로 디코딩될 때, "bin"(주파수 인덱스)이 가장 빠르게 증가하는 인덱스이고, "g"가 가장 느리게 증가하는 인덱스이도록 한다.

[0097] 낮은 주파수와 관련된 스펙트럼 계수는 스펙트럼 계수가 높은 주파수와 관련되기 전에 인코딩된다.

[0098] 변환-코딩된-여기(tcx)에서의 계수는 어레이 x_tcx_invquant[win][bin]에 직접 저장되고, 잡음없는 코딩 코드워드의 전송 순서는 이들이 어레이에 수신되고 저장된 순서로 디코딩될 때, "bin"이 가장 빠르게 증가하는 인덱스이고, "win"이 가장 느리게 증가하는 인덱스이도록 한다. 환언하면, 스펙트럼 값이 음성 코더의 선형-예측 필터의 변환-코딩된-여기를 나타내면, 스펙트럼 값 a은 변환-코딩된-여기의 인접한 및 증가하는 주파수와 관련된다.

[0099] 낮은 주파수에 관련된 스펙트럼 계수는 스펙트럼 계수가 높은 주파수와 관련되기 전에 인코딩된다.

[0100] 특히, 주파수-도메인 대 시간-도메인 신호 변환을 이용하는 시간-도메인 오디오 신호 표현의 "직접" 생성 및, 주파수-도메인 대 시간-도메인 디코더 및 주파수-도메인 대 시간-도메인 신호 변환기의 출력에 의해 여기되는 선형-예측-필터의 양방을 이용하는 오디오 신호 표현의 "간접" 제공의 양방을 위해, 오디오 디코더(200)는 산술 디코더(230)에 의해 제공되는 디코딩된 주파수-도메인 오디오 표현(232)을 적용하도록 구성될 수 있다.

[0101] 환언하면, 여기서 기능이 상세히 논의되는 산술 디코더(200)는, 주파수-도메인으로 인코딩되는 오디오 콘텐츠의 시간-주파수-도메인 표현의 스펙트럼 값의 디코딩 및, 선형-예측-도메인으로 인코딩되는 음성 신호를 디코딩하도록 적응되는 선형-예측-필터에 대한 자극 신호의 시간-주파수-도메인 표현의 제공에 적합하다. 따라서, 산술 디코더는 주파수-도메인-인코딩된 오디오 콘텐츠 및 선형-예측-주파수-도메인-인코딩된 오디오 콘텐츠 (변환-코딩된-여기 선형 예측 도메인 모드)의 양방을 처리할 수 있는 오디오 디코더에 적합하다.

[0102] 6.3. 도 5a 및 5b에 따른 컨텍스트 초기화

[0103] 다음에는, 단계(310)에서 수행되는 컨텍스트 초기화(또한 "컨텍스트 맵핑"으로 명시됨)가 설명될 것이다.

[0104] 컨텍스트 초기화는 도 5a에 도시된 알고리즘 "arith_map_context()"에 따라 과거 컨텍스트와 현재 컨텍스트 사이의 맵핑을 포함한다. 알 수 있듯이, 현재 컨텍스트는 둘 중 1차원 및 n_context의 2차원을 가진 어레이의 형식을 취하는 글로벌 변수 q[2][n_context]에 저장된다. 과거 컨텍스트는 n_context의 차원을 가진 테이블의 형식을 취하는 변수 qs[n_context]에 저장된다. 변수 "previous_lg"는 과거 컨텍스트의 스펙트럼 값의 수를 나타낸다.

[0105] 변수 "lg"는 프레임에서 디코딩할 스펙트럼 계수의 수를 나타낸다. 변수 "previous_lg"는 이전의 프레임의 스펙트럼 라인의 이전의 수를 나타낸다.

[0106] 컨텍스트의 맵핑은 알고리즘 "arith_map_context()"에 따라 수행될 수 있다. 여기서, 현재(예컨대, 주파수-도메인-인코딩된) 오디오 프레임과 관련된 스펙트럼 값의 수가 i=0 내지 i=lg-1에 대한 이전의 오디오 프레임과 관련된 스펙트럼 값의 수와 동일할 경우에, 함수 "arith_map_context()"는 현재 컨텍스트 어레이 q의 엔트리 q[0][i]를 과거 컨텍스트 어레이 qs의 값 qs[i]으로 설정하는 것으로 언급되어야 한다.

[0107] 그러나, 현재 오디오 프레임에 관련된 스펙트럼 값의 수가 이전의 오디오 프레임에 관련된 스펙트럼 값의 수와 상이할 경우에는 더욱 복잡한 맵핑이 수행된다. 그러나, 이 경우에 맵핑에 관한 상세 사항은 특히 본 발명의 핵심 아이디어에 관련이 없어, 상세 사항을 위해 도 5a의 의사 프로그램 코드에 대한 참조가 행해진다.

[0108] 6.4 도 5b 및 5c에 따른 상태 값 계산

[0109] 다음에는. 상태 값 계산(312a)이 더 상세히 설명될 것이다.

[0110] (도 3에 도시된 바와 같은) 제 1 상태 값 s는 함수 "arith_get_context(i, lg, arith_reset_flag, N/2)"의 복귀 값으로 획득될 수 있으며, 이의 의사 프로그램 코드 표현은 도 5b 및 5c에 도시되는 것으로 언급되어야 한

다.

[0111] 상태 값의 계산에 관하여, 또한, 상태 평가에 이용되는 컨텍스트를 도시한 도 4에 대한 참조가 행해진다. 도 4는 시간 및 주파수의 양방에 관한 스펙트럼 값의 2차원 표현을 도시한다. 가로 좌표(410)는 시간을 나타내고, 세로 좌표(412)는 주파수를 나타낸다. 도 4에서 알 수 있듯이. 디코딩하는 스펙트럼 값(420)은 시간 인덱스 t0 및 주파수 인덱스 i와 관련된다. 알 수 있듯이, 시간 인덱스 t0의 경우에, 주파수 인덱스 i-1, i-2 및 i-3을 갖는 튜플(tuples)은 이미 주파수 인덱스 i를 가진 스펙트럼 값(420)이 디코딩될 수 있는 시간에 디코딩된다. 도 2에서 알 수 있듯이, 시간 인덱스 t0 및 주파수 인덱스 i-1를 가진 스펙트럼 값(430)은 이미 스펙트럼 값(420)이 디코딩되기 전에 디코딩되고, 스펙트럼 값(430)은 스펙트럼 값(420)의 디코딩에 이용되는 컨텍스트를 위해 고려된다. 마찬가지로, 시간 인덱스 t0 및 주파수 인덱스 i-2를 가진 스펙트럼 값(434)은 이미 스펙트럼 값(420)이 디코딩되기 전에 디코딩되고, 스펙트럼 값(434)은 스펙트럼 값(420)의 디코딩에 이용되는 컨텍스트를 위해 고려된다. 마찬가지로, 시간 인덱스 t-1 및 주파수 인덱스 i-2를 가진 스펙트럼 값(440), 시간 인덱스 t-1 및 주파수 인덱스 i-1를 가진 스펙트럼 값(444), 시간 인덱스 t-1 및 주파수 인덱스 i를 가진 스펙트럼 값(448), 시간 인덱스 t-1 및 주파수 인덱스 i+1를 가진 스펙트럼 값(452), 및 시간 인덱스 t-1 및 주파수 인덱스 i+2를 가진 스펙트럼 값(456)은 이미 스펙트럼 값(420)이 디코딩되기 전에 디코딩되고, 스펙트럼 값(420)의 디코딩에 이용되는 컨텍스트의 결정을 위해 고려된다. 스펙트럼 값(420)이 디코딩되고, 컨텍스트를 위해 고려되는 시간에 이미 디코딩된 스펙트럼 값(계수)은 음영 사각형으로 표시된다. 이에 반해, (스펙트럼 값(420)이 디코딩될 시에 이미 디코딩되고, 점선을 가진 사각형으로 나타내는 일부 다른 스펙트럼 값, 및 (스펙트럼 값(420)이 디코딩될 시에 아직 디코딩되지 않고, 점선을 가진 원으로 표시되는 다른 스펙트럼 값은 스펙트럼 값(420)을 디코딩하기 위한 컨텍스트를 결정하는데 이용되지 않는다.

[0112] 그러나, 그럼에도 불구하고, 스펙트럼 값(420)을 디코딩하기 위한 컨텍스트"의 정규" (또는 "보통") 계산에 이용되지 않는 이들 스펙트럼 값의 일부는, 개별적으로 또는 종합하여, 크기에 관한 미리 정해진 조건을 충족하는 다수의 이전의 디코딩된 인접 스펙트럼 값의 검출을 위해 평가될 수 있는 것으로 언급되어야 한다.

[0113] 의사 프로그램 코드의 형식으로 함수 "arith_get_context()"의 기능을 도시하는 도 5b 및 5c를 이제 참조하면, 함수 "arith_get_context()"에 의해 수행되는 제 1 컨텍스트 값 "s"의 계산에 관한 약간 더 상세 사항이 설명될 것이다.

[0114] 함수 "arith_get_context()"는 입력 변수로서 디코딩하는 스펙트럼 값의 인덱스 i를 수신하는 것으로 언급되어야 한다. 인덱스 i는 통상적으로 주파수 인덱스이다. 입력 변수 lg는 (현재 오디오 프레임에 대한) 예상된 양자화된 계수의 (전체) 수를 나타낸다. 변수 N은 변환의 라인의 수를 나타낸다. 플래그 "arith_reset_flag"는 컨텍스트가 재설정되어야하는지의 여부를 나타낸다. 함수 "arith_get_context"는 출력 값으로서, 결합(concatenated) 상태 인덱스 s 및 예측된 비트-플레인 레벨 lev0을 나타내는 변수 "t"를 제공한다.

[0115] 함수 "arith_get_context()"는 정수 변수 a0, c0, c1, c2, c3, c4, c5, c6, lev0, 및 "region"를 이용한다.

[0116] 함수 "arith_get_context()"는, 주요 기능 블록으로서, 제 1 산술 재설정 처리(510), 다수의 이전의 디코딩된 인접한 제로 스펙트럼 값의 그룹의 검출(512), 제 1 변수 설정(514), 제 2 변수 설정(516), 레벨 적응(518), 영역 값 설정(520), 레벨 적응(522), 레벨 제한(524), 산술 재설정 처리(526), 제 3 변수 설정(528), 제 4 변수 설정(530), 제 5 변수 설정(532), 레벨 적응(534), 및 선택적 복귀 값 계산(536)을 포함한다.

[0117] 제 1 산술 재설정 처리(510)에서는, 디코딩하는 스펙트럼 값의 인덱스가 제로와 동일하면서, 산술 재설정 플래그 "arith_reset_flag"가 설정되는지의 여부가 검사된다. 이 경우에, 제로의 컨텍스트 값은 복귀되고, 기능은 중단된다.

[0118] 산술 재설정 플래그가 비활성화하고, 디코딩하는 스펙트럼 값의 인덱스 i가 제로와 다를 경우에만 수행되는 다수의 이전의 디코딩된 제로 스펙트럼 값의 그룹의 검출(512)에서, "flag"라 하는 변수는 참조 번호(512a)에 도시된 바와 같이 1로 초기화되고, 평가될 수 있는 스펙트럼 값의 영역은 참조 번호(512b)에 도시된 바와 같이 결정된다. 그 다음, 참조 번호(512b)에 도시된 바와 같이 결정되는 스펙트럼 값의 영역은 참조 번호(512c)에 도시된 바와 같이 평가된다. 이전의 디코딩된 제로 스펙트럼 값의 충분한 영역이 있는 것으로 발견되면, 1의 컨텍스트 값은 참조 번호(512d)에 도시된 바와 같이 복귀된다. 예컨대, 디코딩되는 스펙트럼 값의 인덱스 i가 최대 주파수 인덱스 lg-1에 근접하지 않으면, 상위 주파수 인덱스 경계 "lim_max"는 i+6으로 설정되며, 그런 경우에, 상위 주파수 인덱스 경계의 특정 설정은 참조 번호(512b)에 도시된 바와 같이 행해진다. 더욱이, 디코딩하는 스펙트럼 값의 인덱스 i가 제로에 근접하지 않으면(i+lim_min<0), 하위 주파수 인덱스 경계 "lim_min"는 -5로 설정되며, 그런 경우에, 하위 주파수 인덱스 경계 lim_min의 특정 계산은 참조 번호

(512b)에 도시된 바와 같이 수행된다. 단계(512b)에서 결정된 스펙트럼 값의 영역을 평가할 때, 먼저, 하위 주파수 인덱스 경계 lim_min과 제로 사이의 음의 주파수 인덱스 k에 대한 평가가 수행된다. lim_min과 제로 사이의 주파수 인덱스 k에 대해, 컨텍스트 값 q[0][k].c 및 q[1][k].c 중 적어도 하나는 제로와 동일한지의 여부가 검증된다. 그러나, lim_min과 제로 사이의 어떤 주파수 인덱스 k에 대해 컨텍스트 값 q[0][k].c 및 q[1][k].c의 양방이 제로와 다를 경우, 제로 스펙트럼 값의 충분한 그룹이 없고, 평가(512c)가 중단되는 것으로 결론이 난다. 그 다음, 제로와 lim_max 사이의 주파수 인덱스에 대한 컨텍스트 값 q[0][k].c이 평가된다. 제로와 lim_max 사이의 어떤 주파수 인덱스에 대한 어떤 컨텍스트 값 q[0][k].c이 제로와 다른 것으로 발견되면, 이전의-디코딩된 제로 스펙트럼 값의 충분한 그룹이 없고, 평가(512c)가 중단되는 것으로 결론이 난다. 그러나, lim_min과 제로 사이의 모든 주파수 인덱스 k에 대해 제로와 동일한 적어도 하나의 컨텍스트 값 q[0][k].c 또는 q[1][k].c이 있는 것으로 발견되고, 제로와 lim_max 사이의 모든 주파수 인덱스 k에 대한 제로 컨텍스트 값 q[0][k].c이 있으면, 이전의-디코딩된 제로 스펙트럼 값의 충분한 그룹이 있는 것으로 결론이 난다. 따라서, 이 경우에, 1의 컨텍스트 값은 어떤 추가적 계산 없이 이러한 조건을 나타내도록 복귀된다. 환언하면, 제로의 값을 가진 다수의 컨텍스트 값 q[0][k].c, q[1][k].c의 충분한 그룹이 식별되면, 계산(514, 516, 518, 520, 522, 524, 526, 528, 530, 532, 534, 536)은 스킵된다. 환언하면, 컨텍스트 상태(s)를 나타내는 복귀된 컨텍스트 값은 미리 정해진 조건이 충족되는 검출에 응답하여 이전에 디코딩된 스펙트럼 값과 무관하게 결정된다.

[0119] 그렇지 않으면, 즉, 제로인 컨텍스트 값 q[0][k].c, q[1][k].c의 충분한 그룹이 없으면, 계산(514, 516, 518, 520, 522, 524, 526, 528, 530, 532, 534, 536)의 적어도 일부는 실행된다.

[0120] 디코딩되는 스펙트럼 값의 인덱스 i가 1보다 작을 경우에(및 경우에만) 선택적으로 실행되는 제 1 변수 설정(514)에서, 변수 a_0는 컨텍스트 값 q[1][i-1]을 취하도록 초기화되고, 변수 c0는 변수 a0의 절대값을 취하도록 초기화된다. 변수 "lev0"는 0의 값을 취하도록 초기화된다. 그 다음, 변수 "lev0" 및 c0는 변수 a0가 비교적 큰 절대값을 포함할 경우에, 즉 -4보다 작거나, 4보다 크거나 동일할 경우에 증가된다. 변수 a0의 값이 시프트-대-오른쪽 연산에 의해 -4와 3 사이의 범위로 가져올 때까지 변수 "lev0" 및 c0 변수의 증가는 반복적으로 수행된다(단계 (514b)).

[0121] 그 다음, 변수 c0 및 "lev0"는 제각기 7 및 3의 최대값으로 제한된다(단계 (514c)).

[0122] 디코딩되는 스펙트럼 값의 인덱스 i가 1과 동일하고, 산술 재설정 플래그("arith_reset_flag")가 활성적이면, 변수 c0 및 lev0에만 기초하여 계산되는 컨텍스트 값은 복귀된다. 따라서, 디코딩하는 스펙트럼 값과 같은 시간 인덱스 및, 디코딩되는 스펙트럼 값의 주파수 인덱스 i보다 1만큼 작은 주파수 인덱스를 가진 단일의 이전에 디코딩된 스펙트럼 값만이 컨텍스트 계산을 위해 고려된다(단계 (514d)). 그렇지 않으면, 즉, 산술 재설정 기능이 없는 경우, 변수 c4는 초기화된다(단계 (514e)).

[0123] 결론적으로 말하면, 제 1 변수 설정(514)에서, 변수 c0 및 "lev0"는 이전에 디코딩된 스펙트럼 값에 따라 초기화되고, 현재 디코딩되는 스펙트럼 값과 동일한 프레임과 이전의 스펙트럼 빈(bin) i-1에 대해 디코딩된다. 변수 c4는 이전에 디코딩된 스펙트럼 값에 따라 초기화되고, 현재 디코딩되는 스펙트럼 값과 관련된 주파수보다 (예컨대, 1의 주파수 빈만큼) 낮은 주파수(및 시간 인덱스 t-1)를 가진 이전의 오디오 프레임에 대해 디코딩된다.

[0124] 현재 디코딩되는 스펙트럼 값의 주파수 인덱스가 1보다 클 경우에(및 경우에만) 선택적으로 실행되는 제 2 변수 설정(516)은 변수 c1 및 c6의 초기화 및 변수 lev0의 업데이트를 포함한다. 변수 c1은 현재 오디오 프레임의 이전에 디코딩된 스펙트럼 값과 관련된 컨텍스트 값 q[1][i-2].c에 따라 업데이트되고, 이의 주파수는 현재 디코딩되는 스펙트럼 값의 주파수보다 (예컨대, 2의 주파수 빈만큼) 작다. 마찬가지로, 변수 c6은 (시간 인덱스 t-1를 가진) 이전의 프레임의 이전에 디코딩된 스펙트럼 값을 나타내는 컨텍스트 값 q[0][i-2].c에 따라 초기화되고, 이의 관련된 주파수는 현재 디코딩되는 스펙트럼 값과 관련된 주파수보다 (예컨대, 2의 주파수 빈만큼) 작다. 게다가, 레벨 변수 "lev0"는 현재 프레임의 이전에 디코딩된 스펙트럼 값과 관련된 레벨 값 q[1][i-2].l로 설정되고, 이의 관련된 주파수는, q[1][i-2].l이 lev0보다 클 경우에, 현재 디코딩되는 스펙트럼 값과 관련된 주파수보다 (예컨대, 2의 주파수 빈만큼) 작다.

[0125] 레벨 적응(518) 및 영역 값 설정(520)은 디코딩되는 스펙트럼 값의 인덱스 i가 2보다 클 경우에(및 경우에만) 선택적으로 실행된다. 레벨 적응(518)에서, 관련된 주파수가 현재 디코딩되는 스펙트럼 값과 관련된 주파수보다 (예컨대, 3의 주파수 빈만큼) 작은 현재 프레임의 이전에 디코딩된 스펙트럼 값에 관련되는 레벨 값 q[1][i-3].l이 레벨 값 lev0보다 클 경우에, 레벨 변수 "lev0"는 q[1][i-3].l의 값으로 증가된다.

[0126] 영역 값 설정(520)에서, 변수 "region"는 평가에 따라 설정되며, 다수의 스펙트럼 영역 중 어떤 스펙트럼 영역에서, 현재 디코딩되는 스펙트럼 값이 배치된다. 예컨대, 현재 디코딩되는 스펙트럼 값이 주파수 빈(0 ≤ i < N/4)의 제 1(최하위) 쿼터 내에 있는 (주파수 빈 인덱스 i를 갖는) 주파수 빈에 관련되어 있는 것으로 발견되면, 영역 변수 "region"는 0으로 설정된다. 그렇지 않으면, 현재 디코딩되는 스펙트럼 값이 현재 프레임에 관련된 주파수 빈(N/4 ≤ i < N/2)의 제 2 쿼터 내에 있는 주파수 빈에 관련되면, 영역 변수는 1의 값으로 설정된다. 그렇지 않으면, 즉, 현재 디코딩되는 스펙트럼 값이 주파수 빈(N/2 ≤ i < N)의 제 2 (상위) 절반 내에 있는 주파수 빈에 관련되면, 영역 변수는 2로 설정된다. 따라서, 영역 변수는 현재 디코딩되는 스펙트럼 값이 관련되는 어떤 주파수 영역에 대한 평가에 따라 설정된다. 2 이상의 주파수 영역이 구분될 수 있다.

[0127] 추가적 레벨 적응(522)은 현재 디코딩되는 스펙트럼 값이 3보다 큰 스펙트럼 인덱스를 포함할 경우에(및 경우에만) 실행된다. 이 경우에, 현재 디코딩되는 스펙트럼 값에 관련된 주파수보다, 예컨대, 4의 주파수 빈만큼 작은 주파수에 관련되는 현재 프레임의 이전에 디코딩된 스펙트럼 값에 관련되는 레벨 값 q[1][i-4].l이 현재 레벨 "lev0"보다 클 경우에 레벨 변수 "lev0"는 증가된다(value q[1][i-4].l으로 설정된다)(단계 (522)). 레벨 변수 "lev0"는 3의 최대값으로 제한된다(단계 (524)).

[0128] 산술 재설정 조건이 검출되고, 현재 디코딩되는 스펙트럼 값의 인덱스 i가 1보다 큰 경우, 상태 값은 변수 c0, c1, lev0 뿐만 아니라 영역 변수 "region"에 따라 복귀된다(단계 (526)). 따라서, 어떤 이전의 프레임의 이전에 디코딩된 스펙트럼 값은 산술 재설정 조건이 주어질 경우에 무시된다.

[0129] 제 3 변수 설정(528)에서, 변수 c2는 (시간 인덱스 t-1를 가진) 이전의 오디오 프레임의 이전에 디코딩된 스펙트럼 값에 관련되는 컨텍스트 값 q[0][i].c으로 설정되고, 이전에 디코딩된 스펙트럼 값은 현재 디코딩되는 스펙트럼 값과 동일한 주파수와 관련된다.

[0130] 제 4 변수 설정(530)에서, 현재 디코딩되는 스펙트럼 값이 최고 가능한 주파수 인덱스 lg-1와 관련되지 않으면, 변수 c3는 주파수 인덱스 i+1를 가진 이전의 오디오 프레임의 이전에 디코딩된 스펙트럼 값에 관련되는 컨텍스트 값 q[0][i+1].c으로 설정된다.

[0131] 제 5 변수 설정(532)에서, 현재 디코딩되는 스펙트럼 값의 주파수 인덱스 i가 최대 주파수 인덱스 값에 너무 가깝지 않으면(즉, 주파수 인덱스 값 lg-2 또는 lg-1을 취하지 않으면) 변수 c5는 주파수 인덱스 i+2를 가진 이전의 오디오 프레임의 이전에 디코딩된 스펙트럼 값에 관련되는 컨텍스트 값 q[0][i+2].c으로 설정된다.

[0132] 레벨 변수 "lev0"의 추가적 적응은 주파수 인덱스 i가 0과 동일할 경우에(즉, 현재 디코딩되는 스펙트럼 값이 최저 스펙트럼 값인 경우에) 수행된다. 이 경우에, 현재 인코딩되는 스펙트럼 값과 관련된 주파수에 비해, 변수 c2 또는 c3가 동일한 주파수 또는 심지어 더 높은 주파수와 관련되는 이전의 오디오 프레임의 이전에 디코딩된 스펙트럼 값이 비교적 큰 값을 취하는 것을 나타내는 3의 값을 취할 경우에 레벨 변수 "lev0"는 0에서 1로 증가된다.

[0133] 선택적 복귀 값 계산(536)에서, 복귀 값은 현재 디코딩되는 스펙트럼 값의 인덱스 i가 값 0, 1, 또는 더 큰 값을 취하는지의 여부에 따라 계산된다. 인덱스 i가 0의 값을 취할 경우에 참조 번호(536a)에 나타낸 바와 같이, 복귀 값은 변수 c2, c3, c5 및 lev0에 따라 계산된다. 인덱스 i가 1의 값을 취할 경우에 참조 번호(536b)에 도시된 바와 같이, 복귀 값은 변수 c0, c2, c3, c4, c5, 및 "lev0"에 따라 계산된다. 인덱스 i가 0 또는 1과 상이한 값을 취할 경우에(참조 번호(536c)), 복귀 값은 변수 c0, c2, c3, c4, c1, c5, c6, "region", 및 lev0에 따라 계산된다.

[0134] 상술한 바를 요약하면, 컨텍스트 값 계산 "arith_get_context()"은 다수의 이전의 디코딩된 제로 스펙트럼 값(또는 적어도, 상당히 작은 스펙트럼 값)의 그룹의 검출(512)을 포함한다. 이전의 디코딩된 제로 스펙트럼 값의 충분한 그룹이 발견되면, 특정 컨텍스트의 존재는 복귀 값을 1로 설정하여 나타낸다. 그렇지 않으면, 컨텍스트 값 계산이 수행된다. 일반적으로, 컨텍스트 값 계산에서, 인덱스 값 i은 얼마나 많은 이전의 디코딩된 스펙트럼 값이 평가되어야 하는 지를 판단하기 위해 평가된다고 할 수 있다. 예컨대, 평가된 이전의 디코딩된 스펙트럼 값의 수는 현재 디코딩되는 스펙트럼 값의 주파수 인덱스 i가 하위 경계(예컨대, 0)에 가깝거나, 상위 경계(예컨대, lg-1)에 가까울 경우에 감소된다. 게다가, 현재 디코딩되는 스펙트럼 값의 주파수 인덱스 i가 최소값에서 상당히 떨어질지라도, 서로 다른 스펙트럼 영역은 영역 값 설정(520)에 의해 구별된다. 따라서, 서로 다른 스펙트럼 영역(예컨대, 제 1 낮은 주파수 스펙트럼 영역, 제 2 중간 주파수 스펙트럼 영역, 및 제 3 높은 주파수 스펙트럼 영역)의 서로 다른 통계적 특성이 고려된다. 복귀 값으로 계산되는 컨텍스트 값은, 현재 디코딩되는 스펙트럼 값이 제 1 미리 정해진 주파수 영역 또는 제 2 미리 정해진 주파수 영역 (또는 어떤 다른 미리 정해진 주파수 영역)에 있는지에 복귀된 컨텍스트 값이 의존하도록 변수 "region"에

의존한다.

[0135] 6.5 맵핑 규칙 선택

[0136] 다음에는, 맵핑 규칙의 선택, 예컨대, 심볼 코드로의 코드 값의 맵핑을 나타내는 누적-빈도-테이블이 설명된다. 맵핑 규칙의 선택은 상태 값 s 또는 t에 의해 나타내는 컨텍스트 상태에 따라 행해진다.

[0137] 6.5.1 도 5d에 따른 알고리즘을 이용하는 맵핑 규칙 선택

[0138] 다음에는, 도 5d에 따른 함수 "get_pk"를 이용한 맵핑 규칙의 선택이 설명될 것이다. 함수 "get_pk"는 도 3의 알고리즘의 서브알고리즘(312ba)에서 "pki"의 값을 획득하기 위해 수행될 수 있는 것으로 언급되어야 한다. 따라서, 함수 "get_pk"는 도 3의 알고리즘에서의 함수 "arith_get_pk"를 대신할 수 있다.

[0139] 또한, 도 5d에 따른 함수 "get_pk"는 도 17a 및 17b에 따른 테이블 "ari_s_hash[387]" 및, 도 18에 따른 테이블 "ari_gs_hash"[225]을 평가할 수 있다.

[0140] 함수 "get_pk"는, 입력 변수로서, 도 3에 따른 변수 "t" 및 도 3에 따른 변수 "lev", "lev0"의 조합에 의해 획득될 수 있는 상태 값 s을 받는다. 함수 "get_pk"는 또한, 복귀 값으로서, 맵핑 규칙 또는 누적-빈도-테이블을 명시하는 변수 "pki"의 값을 복귀시키도록 구성된다. 함수 "get_pk"는 상태 값 s을 맵핑 규칙 인덱스 값 "pki"으로 맵핑하도록 구성된다.

[0141] 함수 "get_pk"는 제 1 테이블 평가(540) 및 제 2 테이블 평가(544)를 포함한다. 제 1 테이블 평가(540)는, 참조 번호(541)에 도시된 바와 같이, 변수 i_min, i_max, 및 i가 초기화되는 변수 초기화(541)를 포함한다. 제 1 테이블 평가(540)는 또한 상태 값 s과 일치하는 테이블 "ari_s_hash"의 엔트리(entry)가 있는지의 여부에 관해 결정이 행해지는 동안에 반복 테이블 탐색(542)을 포함한다. 이와 같은 일치가 반복 테이블 탐색(542) 동안에 식별되면, 함수 get_pk는 중단되며, 함수의 복귀 값은, 더욱 상세히 설명되는 바와 같이, 상태 값 s과 일치하는 테이블 "ari_s_hash"의 엔트리에 의해 결정된다. 그러나, 상태 값 s과 테이블 "ari_s_hash"의 엔트리 사이의 완전한 일치가 반복 테이블 탐색(542) 동안에 발견되지 않으면, 경계 엔트리 검사(543)가 수행된다.

[0142] 이제 제 1 테이블 평가(540)의 상세 사항에 의하면, 탐색 구간은 변수 i_min 및 i_max에 의해 정의되는 것으로 볼 수 있다. 반복 테이블 탐색(542)은 변수 i_min 및 i_max에 의해 정의되는 구간이 충분히 크고, 조건 i_max-i_min > 1이 충족될 수 있기만 하면 반복된다. 그 다음, 변수 i는, 적어도 대략, 구간 (i=i_min+(i_max-i_min)/2)의 중간을 명시하도록 설정된다. 그 다음, 변수 j는 변수 i에 의해 명시되는 어레이 위치에서 어레이 "ari_s_hash"에 의해 결정되는 값으로 설정된다(참조 번호(542)). 여기서, 테이블 "ari_s_hash"의 각 엔트리는 테이블 엔트리에 관련되는 상태 값, 및 테이블 엔트리에 관련되는 맵핑 규칙 인덱스 값의 양방을 나타내는 것으로 언급되어야 한다. 테이블 엔트리에 관련되는 상태 값은 테이블 엔트리의 상위 비트(비트 8-31)에 의해 나타내지만, 맵핑 규칙 인덱스 값은 상기 테이블 엔트리의 하위 비트(예컨대, 비트 0-7)에 의해 나타낸다. 하위 경계 i_min 또는 상위 경계 i_max는 상태 값 s이 변수 i에 의해 참조되는 테이블 "ari_s_hash"의 엔트리 "ari_s_hash[i]"의 최상위 24 비트에 의해 나타내는 상태 값보다 작은지의 여부에 따라 적응된다. 예컨대, 상태 값 s이 엔트리 "ari_s_hash[i]"의 최상위 24 비트에 의해 나타내는 상태 값보다 작으면, 테이블 구간의 상위 경계 i_max는 값 i로 설정된다. 따라서, 반복 테이블 탐색(542)의 다음 반복을 위한 테이블 구간은 반복 테이블 탐색(542)의 현재 반복에 이용되는 테이블 구간 (i_min에서 i_max까지)의 하위 절반으로 제한된다. 반면에, 상태 값 s이 테이블 엔트리 "ari_s_hash[i]"의 최상위 24 비트에 의해 나타내는 상태 값보다 크면, 반복 테이블 탐색(542)의 다음 반복을 위한 테이블 구간의 하위 경계 i_min는 값 i로 설정되어, (i_min와 i_max 사이의) 현재 테이블 구간의 상위 절반이 반복 테이블 탐색을 위한 테이블 구간으로 이용되도록 한다. 그러나, 상태 값 s이 테이블 엔트리 "ari_s_hash[i]"의 최상위 24 비트에 의해 나타내는 상태 값과 동일한 것으로 발견되면, 테이블 엔트리 "ari_s_hash[i]"의 최하위 8 비트에 의해 나타내는 맵핑 규칙 인덱스 값은 "함수 "get_pk"에 의해 복귀되고, 함수는 중단된다.

[0143] 반복 테이블 탐색(542)은 변수 i_min 및 i_max에 의해 정의된 테이블 구간이 충분히 작을 때까지 반복된다.

[0144] 경계 엔트리 검사(543)는 (선택적으로) 반복 테이블 탐색(542)을 보충하기 위해 실행된다. 인덱스 변수 i가 반복 테이블 탐색(542)의 완료 후에 인덱스 변수 i_max와 같다면, 상태 값 s이 테이블 엔트리 "ari_s_hash[i_min]"의 최상위 24 비트에 의해 나타내는 상태 값과 동일한지 최종 검사가 행해지고, 테이블 엔트리 "ari_s_hash[i_min]"의 최하위 8 비트에 의해 나타내는 맵핑 규칙 인덱스 값은 이 경우에 "함수 "get_pk"의 결과로서 복귀된다. 반면에, 인덱스 변수 i가 인덱스 변수 i_max와 상이하면, 상태 값 s이 테이블

엔트리 "ari_s_hash[i_max]"의 최상위 24 비트에 의해 나타내는 상태 값과 동일한지에 관해 검사가 수행되고, 상기 테이블 엔트리 "ari_s_hash[i_max]"의 최하위 8 비트에 의해 나타내는 맵핑 규칙 인덱스 값은 이 경우에 "함수 "get_pk"의 복귀 값으로 복귀된다.

[0145] 그러나, 경계 엔트리 검사(543)는 전체적으로 선택적 사항으로 간주될 수 있는 것으로 언급되어야 한다.

[0146] 제 1 테이블 평가(540) 다음에, "direct hit"가 제 1 테이블 평가(540) 동안에 발생하지 않으면, 상태 값 s이 테이블 "ari_s_hash"의 엔트리(또는, 더욱 정확하게는, 이의 24 최상위 비트)에 의해 나타내는 상태 값 중 하나와 동일하다는 점에서 제 2 테이블 평가(544)가 수행된다.

[0147] 제 2 테이블 평가(544)는, 참조 번호(545)에 도시된 바와 같이, 인덱스 변수 i_min, i 및 i_max가 초기화되는 변수 초기화(545)를 포함한다. 제 2 테이블 평가(544)는 또한 테이블 "ari_gs_hash"이 상태 값 s과 동일한 상태 값을 나타내는 엔트리를 위해 탐색되는 동안에 반복 테이블 탐색(546)을 포함한다. 최종으로, 제 2 테이블 탐색(544)은 복귀 값 결정(547)을 포함한다.

[0148] 반복 테이블 탐색(546)은 인덱스 변수 i_min 및 i_max에 의해 정의되는 테이블 구간이 충분히 크기만 하면(예컨대, i_max - i_min > 1이기만 하면) 반복된다. 반복 테이블 탐색(546)의 반복에서, 변수 i는 i_min 및 i_max에 의해 정의된 테이블 구간의 중심으로 설정된다(단계 (546a)). 그 다음, 테이블 "ari_gs_hash"의 엔트리 j는 인덱스 변수 i에 의해 결정되는 테이블 위치에서 획득된다(546b). 환언하면, 테이블 엔트리 "ari_gs_hash[i]"는 테이블 인덱스 i_min 및 i_max에 의해 정의되는 현재 테이블 구간의 중심에 있는 테이블 엔트리이다. 그 다음, 반복 테이블 탐색(546)의 다음 반복을 위한 테이블 구간이 결정된다. 이를 위해, 테이블 구간의 상위 경계를 나타내는 인덱스 값 i_max은, 상태 값 s이 테이블 엔트리 "j=ari_gs_hash[i]"의 최상위 24 비트에 의해 나타내는 상태 값보다 작을 경우에 값 i으로 설정된다(546c). 환언하면, 현재 테이블 구간의 하위 절반은 반복 테이블 탐색(546)의 다음 반복을 위한 새로운 테이블 구간으로 선택된다(단계 (546c)). 그렇지 않으면, 상태 값 s이 테이블 엔트리 "j=ari_gs_hash[i]"의 최상위 24 비트에 의해 나타내는 상태 값보다 클 경우, 인덱스 값 i_min은 값 i으로 설정된다. 따라서, 현재 테이블 구간의 상위 절반은 반복 테이블 탐색(546)의 다음 반복을 위한 새로운 테이블 구간으로 선택된다(단계 (546d)). 그러나, 상태 값 s이 테이블 엔트리 "j=ari_gs_hash[i]"의 최상위 24 비트에 의해 나타내는 상태 값과 동일한 것으로 발견되면, 인덱스 변수 i_max는 값 i+1 또는 (i+1이 224보다 클 경우에) 값 224로 설정되며, 반복 테이블 탐색(546)은 중단된다. 그러나, 상태 값 s이 "j=ari_gs_hash[i]"의 최상위 24 비트에 의해 나타내는 상태 값과 상이할 경우, 테이블 구간이 너무 작지 않으면(i_max - i_min ≤ 1) 반복 테이블 탐색(546)은 업데이트된 인덱스 값 i_min 및 i_max에 의해 정의되는 새로이 설정된 테이블 구간으로 반복된다. 따라서, (i_min 및 i_max에 의해 정의되는) 테이블 구간의 구간 크기는 "direct hit"가 검출되거나 (s==(j>>8)), 구간이 최소 허용 가능한 크기(i_max - i_min ≤ 1)에 도달할 때까지 반복적으로 감소된다. 최종으로, 반복 테이블 탐색(546)의 중단에 따라, 테이블 엔트리 "j=ari_gs_hash[i_max]"가 결정되고, 상기 테이블 엔트리 "j=ari_gs_hash[i_max]"의 8 최하위 비트에 의해 나타내는 맵핑 규칙 인덱스 값은 "함수 "get_pk"의 복귀 값으로 복귀된다. 따라서, 맵핑 규칙 인덱스 값은 반복 테이블 탐색(546)의 완료 또는 중단 후에 (i_min 및 i_max에 의해 정의되는) 테이블 구간의 상위 경계 i_max에 따라 결정된다.

[0149] 양자 모두 반복 테이블 탐색(542, 546)을 이용하는 상술한 테이블 평가(540, 544)는 매우 높은 계산 효율을 가진 주어진 중요한 상태의 존재에 대해 테이블 "ari_s_hash" 및 "ari_gs_hash"의 조사를 허용한다. 특히, 테이블 액세스 연산의 수는 최악의 경우에도 알맞게 유지될 수 있다. 테이블 "ari_s_hash" 및 "ari_gs_hash"의 숫자 순서(numeric ordering)는 적절한 해시 값에 대한 탐색의 가속을 허용하는 것으로 발견되었다. 게다가, 테이블 크기는 테이블 "ari_s_hash" 및 "ari_gs_hash"에서의 이스케이프 심볼의 포함(inclusion)이 필요치 않음에 따라 작게 유지될 수 있다. 따라서, 다수의 서로 다른 상태가 있을지라도 효율적인 컨텍스트 해싱 메커니즘이 확립된다: 제 1 스테이지(제 1 테이블 평가(540))에서, 다이렉트 히트(direct hit)에 대한 탐색이 실시된다(s==(j>>8)) .

[0150] 제 2 스테이지(제 2 테이블 평가(544))에서, 상태 값 s의 범위는 맵핑 규칙 인덱스 값으로 맵핑될 수 있다. 따라서, 테이블 "ari_s_hash"에서 관련된 엔트리가 있는 특히 중요한 상태, 및 범위 기반 처리가 있는 덜 중요한 상태의 균형잡힌 처리는 수행될 수 있다. 따라서, 함수 "get_pk"는 맵핑 규칙 선택의 효율적인 구현을 구성한다.

[0151] 어떤 추가적 상세 사항을 위해, 잘 알려진 프로그래밍 언어 C에 따른 표현의 함수 "get_pk"의 기능을 나타내는 도 5d의 의사 프로그램 코드에 대한 참조가 행해진다.

[0152] 6.5.2 도 5e에 따른 알고리즘을 이용하는 맵핑 규칙 선택

[0153] 다음에는, 맵핑 규칙의 선택을 위한 다른 알고리즘이 도 5e를 참조로 설명될 것이다. 도 5e에 따른 알고리즘 "arith_get_pk"는, 입력 변수로서, 컨텍스트의 상태를 나타내는 상태 값 s을 받는다. 함수 "arith_get_pk"는, 출력 값 또는 복귀 값으로서, 맵핑 규칙(예컨대, 누적-빈도-테이블)을 선택하기 위한 인덱스일 수 있는 확률 모델의 인덱스 "pki"를 제공한다.

[0154] 도 5e에 따른 함수 "arith_get_pk"는 도 3의 함수 "value_decode"의 함수 "arith_get_pk"의 기능을 취할 수 있는 것으로 언급되어야 한다.

[0155] 또한, 함수 "arith_get_pk"는, 예컨대, 도 20에 따른 테이블 ari_s_hash 및 도 18에 따른 테이블 ari_gs_hash을 평가할 수 있는 것으로 언급되어야 한다.

[0156] 도 5e에 따른 함수 "arith_get_pk"는 제 1 테이블 평가(550) 및 제 2 테이블 평가(560)를 포함한다. 제 1 테이블 평가(550)에서, 상기 테이블의 엔트리 j=ari_s_hash[i]를 획득하기 위해 테이블 ari_s_hash을 통해 선형 스캔이 행해진다. 테이블 ari_s_hash의 테이블 엔트리 j=ari_s_hash[i]의 최상위 24 비트에 의해 나타내는 상태 값이 상태 값 s과 동일하면, 상기 식별된 테이블 엔트리 j=ari_s_hash[i]의 최하위 8 비트에 의해 나타내는 맵핑 규칙 인덱스 값 "pki"은 복귀되고, 함수 "arith_get_pk"는 중단된다. 따라서, "direct hit"(테이블 엔트리의 최상위 24 비트에 의해 나타내는 상태 값과 동일한 상태 값 s)이 식별되지 않으면, 테이블 ari_s_hash의 모든 387 엔트리는 오름차순 시퀀스로 평가된다.

[0157] 다이렉트 히트가 제 1 테이블 평가(550) 내에서 식별되지 않으면, 제 2 테이블 평가(560)가 실행된다. 제 2 테이블 평가(560) 동안에, 0에서 224의 최대값까지 선형적으로 증가하는 엔트리 인덱스를 가진 선형 스캔이 수행된다. 제 2 테이블 평가 동안, 테이블 i에 대한 테이블 "ari_gs_hash"의 엔트리 "ari_gs_hash[i]"는 판독되고, 테이블 엔트리 "j=ari_gs_hash[i]"는 테이블 엔트리 j의 24 최상위 비트에 의해 나타내는 상태 값이 상태 값 s보다 큰지의 여부가 결정되는 것으로 평가된다. 크다면, 상기 테이블 엔트리 j의 8 최하위 비트에 의해 나타내는 맵핑 규칙 인덱스 값은 함수 "arith_get_pk"의 복귀 값으로서 복귀되고, 함수 "arith_get_pk"의 실행은 중단된다.

[0158] 그러나, 상태 값 s이 현재 테이블 엔트리 "j=ari_gs_hash[i]"의 24 최상위 비트에 의해 나타내는 상태 값보다 작지 않으면, 테이블 ari_gs_hash의 엔트리를 통한 스캔은 테이블 인덱스 i를 증가시킴으로써 계속된다. 그러나, 상태 값 s이 테이블 ari_gs_hash의 엔트리에 의해 나타내는 어떤 상태 값 이상이면, 테이블 ari_gs_hash의 마지막 엔트리의 8 최하위 비트에 의해 정의되는 맵핑 규칙 인덱스 값 "pki"은 함수 "arith_get_pk"의 복귀 값으로서 복귀된다.

[0159] 요약하면, 도 5e에 따른 함수 "arith_get_pk"는 2-단계 해싱을 수행한다. 제 1 단계에서, 다이렉트 히트에 대한 탐색이 수행되며, 여기서, 상태 값 s이 제 1 테이블 "ari_s_hash"의 어떤 엔트리에 의해 정의되는 상태 값과 동일한지의 여부가 결정된다. 다이렉트 히트가 제 1 테이블 평가(550)에서 식별되면, 복귀 값은 제 1 테이블 "ari_s_hash"에서 획득되고, 함수 "arith_get_pk"는 중단된다. 그러나, 다이렉트 히트가 제 1 테이블 평가(550)에서 식별되지 않으면, 제 2 테이블 평가(560)가 수행된다. 제 2 테이블 평가에서, 범위 기반 평가가 수행된다. 제 2 테이블 "ari_gs_hash"의 후속 엔트리가 범위를 정의한다. 상태 값이 (현재 테이블 엔트리 "j=ari_gs_hash[i]"의 24 최상위 비트에 의해 나타내는 상태 값이 상태 값 s보다 크다는 사실에 의해 나타내는) 이와 같은 범위 내에 있는 것으로 발견되면, 테이블 엔트리 j=ari_gs_hash[i]의 8 최하위 비트에 의해 나타내는 맵핑 규칙 인덱스 값 "pki"은 복귀된다.

[0160] 6.5.3 도 5f에 따른 알고리즘을 이용하는 맵핑 규칙 선택

[0161] 도 5f에 따른 함수 "get_pk"는 도 5e에 따른 함수 "arith_get_pk"에 실질적으로 상응한다. 따라서, 상기 논의에 대한 참조가 행해진다. 추가적 상세 사항을 위해, 도 5f의 의사 프로그램 표현에 대한 참조가 행해진다.

[0162] 도 5f에 따른 함수 "get_pk"는 도 3의 함수 "value_decode"로 불리는 함수 "arith_get_pk"를 대신할 수 있는 것으로 언급되어야 한다.

[0163] 6.6. 도 5g에 따른 함수 "arith_decode()"

[0164] 다음에는, 함수 "arith_decode()"의 기능이 도 5g를 참조로 상세히 논의될 것이다. 함수 "arith_decode()"는, 달리 시퀀스 및 FALSE의 제 1 심볼인 경우에 TRUE를 복귀시키는 헬퍼 함수(helper function) "arith_first_symbol (void)"를 이용하는 것으로 언급되어야 한다. 함수 "arith_decode()"는 또한 비트스트림

의 다음 비트를 획득하여 제공하는 헬퍼 함수 "arith_get_next_bit(void)"를 이용한다.

[0165] 게다가, 함수 "arith_decode()"는 글로벌 변수 "low", "high" 및 "value"를 이용한다. 더우기, 함수 "arith_decode()"는, 입력 변수로서, 선택된 누적-빈도-테이블의 (요소 인덱스 또는 엔트리 인덱스 0을 갖는) 제 1 엔트리 또는 요소를 향해 가리키는 변수 "cum_freq[]"를 받는다. 또한, 함수 "arith_decode()"는 변수 "cum_freq[]"에 의해 명시되는 선택된 누적-빈도-테이블의 길이를 나타내는 입력 변수 "cfl"를 이용한다.

[0166] 함수 "arith_decode()"는, 제 1 단계로서, 헬퍼 함수 "arith_first_symbol()"가 심볼의 시퀀스의 제 1 심볼을 디코딩함을 나타낼 경우에 수행되는 변수 초기화(570a)를 포함한다. 값 초기화(550a)는 헬퍼 함수 "arith_get_next_bit"를 이용하여 비트스트림에서 획득되는, 예컨대, 다수의 20 비트에 따라 변수 "value"를 초기화하여, 변수 "value"가 상기 비트에 의해 나타낸 값을 취하도록 한다. 또한, 변수 "low"는 0의 값을 취하도록 초기화되며, 변수 "high"는 1048575의 값을 취하도록 초기화된다.

[0167] 제 2 단계(570b)에서, 변수 "range"는 변수 "high" 및 "low"의 값 사이의 차이보다 1만큼 큰 값으로 설정된다. 변수 "cum"는 변수 "low"의 값과 변수 "high"의 값 사이의 변수 "value"의 값의 상대 위치를 나타내는 값으로 설정된다. 따라서, 변수 "cum"는, 예컨대, 변수 "value"의 값에 따라 0과 2^{16} 사이의 값을 취한다.

[0168] 포인터 p는 선택된 누적-빈도-테이블의 시작 어드레스보다 1만큼 작은 값으로 초기화된다.

[0169] 알고리즘 "arith_decode()"은 반복 누적-빈도-테이블-탐색(570c)을 포함한다. 반복 누적-빈도-테이블-탐색은 변수 cfl가 1 이하일 때까지 반복된다. 반복 누적-빈도-테이블-탐색(570c)에서, 포인터 변수 q는 포인터 변수 p의 현재 값과 변수 "cfl"의 값의 절반의 합과 동일한 값으로 설정된다. 엔트리가 포인터 변수 q에 의해 어드레스되는 선택된 누적-빈도-테이블의 엔트리 *q의 값이 변수 "cum"의 값보다 큰 경우, 포인터 변수 p는 포인터 변수 q의 값으로 설정되고, 변수 "cfl"는 증가된다. 최종으로, 변수 "cfl"는 1 비트씩 오른쪽으로 시프트되어, 변수 "cfl"의 값을 2로 효과적으로 나누고, 모듈로(modulo) 부분을 무시한다.

[0170] 따라서, 반복 누적-빈도-테이블-탐색(570c)은, 누적-빈도-테이블의 엔트리로 묶여있는 선택된 누적-빈도-테이블 내의 구간을 식별하여, 값 cum이 식별된 구간 내에 있도록 하기 위해 선택된 누적-빈도-테이블의 다수의 엔트리와 변수 "cum"의 값을 효과적으로 비교한다. 따라서, 선택된 누적-빈도-테이블의 엔트리는 구간을 정의하고, 각각의 심볼 값은 선택된 누적-빈도-테이블의 구간의 각각과 관련된다. 또한, 누적-빈도-테이블의 두 인접한 값 사이의 구간의 폭은 상기 구간과 관련된 심볼의 확률을 정의하여, 선택된 누적-빈도-테이블이 전체적으로 서로 다른 심볼(또는 심볼 값)의 확률 분포를 정의하도록 한다. 이용 가능한 누적-빈도-테이블에 관한 상세 사항은 도 19를 참조로 아래에서 논의된다.

[0171] 도 5g를 다시 참조하면, 심볼 값은 포인터 변수 p의 값에서 유도되며, 심볼 값은 참조 번호(570d)에 도시된 바와 같이 유도된다. 따라서, 포인터 변수 p의 값과 시작 어드레스 "cum_freq"의 차이는 변수 "symbol"로 나타내는 심볼 값을 획득하기 위해 평가된다.

[0172] 알고리즘 "arith_decode"은 또한 변수 "high" 및 "low"의 적응(570e)을 포함한다. 변수 "symbol"로 나타내는 심볼 값이 0과 다르면, 변수 "high"는 참조 번호(570e)에 도시된 바와 같이 업데이트된다. 또한, 변수 "low"의 값은 참조 번호(570e)에 도시된 바와 같이 업데이트된다. 변수 "high"는 변수 "low", 변수 "range" 및, 선택된 누적-빈도-테이블의 인덱스 "symbol-1"를 가진 엔트리의 값에 의해 결정되는 값으로 설정된다. 변수 "low"는 증가되며, 여기서, 증가의 정도는 변수 "range" 및, 인덱스 "symbol"를 가진 선택된 누적-빈도-테이블의 엔트리에 의해 결정된다. 따라서, 변수 "low" 및 "high"의 값 사이의 차이는 선택된 누적-빈도-테이블의 두 인접한 엔트리 사이의 숫자 차이에 따라 조정된다.

[0173] 따라서, 낮은 확률을 가진 심볼 값이 검출되면, 변수 "low" 및 "high"의 값 사이의 구간은 좁은 폭으로 축소된다. 이에 반해, 검출된 심볼 값이 비교적 큰 확률을 포함하면, 변수 "low" 및 "high"의 값 사이의 구간의 폭은 비교적 큰 값으로 설정된다.

[0174] 다시 말하면, 변수 "low" 및 "high"의 값 사이의 구간의 폭은 검출된 심볼 및 누적-빈도-테이블의 상응하는 엔트리에 의존한다.

[0175] 알고리즘 "arith_decode()"은 또한 단계(570e)에서 결정되는 구간이 반복적으로 시프트되어, "break" 조건이 도달될 때까지 스케일링되는 구간 재규격화(renormalization)(570f)를 포함한다. 구간 재규격화(570f)에서, 선택적 시프트-하향 연산(570fa)이 수행된다. 변수 "high"가 524286보다 작으면, 아무것도 행해지지 않으며,

구간 재규격화는 구간-크기-증가 연산(570fb)으로 계속된다. 그러나, 변수 "high"가 524286보다 작지 않고, 변수 "low가" 524286 이상이면, 변수 "values", "low" 및 "high"는 모두 524286만큼 감소되어, 변수 "low" 및 "high"에 의해 정의되는 구간이 하향으로 시프트되고, 변수 "values"의 값이 또한 하향으로 시프트되도록 한다. 그러나, 변수 "high"의 값이 524286보다 작지 않고, 변수 "low가" 524286 이하이며, 변수 "low가" 262143 이상이며, 변수 "high"가 786429보다 작으면, 변수 "values", "low" 및 "high"는 모두 262143만큼 감소되어, 변수 "high" 및 "low"의 값 및 또한 변수 "values"의 값 사이의 구간을 하향 시프트시킨다. 그러나, 상기 조건의 어느 것도 충족되지 않으면, 구간 재규격화가 중단된다.

[0176] 그러나, 단계(570fa)에서 평가되는 상술한 조건 중 어느 하나가 충족되면, 구간-증가-연산(570fb)이 실행된다. 구간-증가-연산(570fb)에서, 변수 "low"의 값은 두 배가 된다. 또한, 변수 "high"의 값은 두 배가 되고, 두 배의 결과는 1씩 증가된다. 또한, 변수 "value"의 값은 두 배가 되고(1 비트씩 왼쪽으로 시프트되고), 헬퍼 함수 "arith_get_next_bit"에 의해 획득되는 비트스트림의 비트는 최하위 비트로서 이용된다. 따라서, 변수 "low" 및 "high"의 값 사이의 구간의 크기는 대략 두 배가 되고, 변수 "value"의 정밀도는 비트스트림의 새로운 비트를 이용함으로써 증가된다. 상술한 바와 같이, 단계(570fa 및 570fb)는 "break" 조건이 도달될 때까지, 즉, 변수 "low" 및 "high"의 값 사이의 구간이 충분히 클 때까지 반복된다.

[0177] 알고리즘 "arith_decode()"의 기능에 관해, 변수 "low" 및 "high"의 값 사이의 구간은, 단계(570e)에서, 변수 "cum_freq"에 의해 참조되는 누적-빈도-테이블의 두 인접한 엔트리에 따라 감소되는 것으로 언급되어야 한다. 선택된 누적-빈도-테이블의 두 인접한 값 사이의 구간이 작으면, 즉, 인접한 값이 비교적 서로 가까우면, 단계(570e)에서 획득되는 변수 "low" 및 "high"의 값 사이의 구간은 비교적 작을 것이다. 이에 반해, 누적-빈도-테이블의 두 인접한 엔트리가 더 이격되면, 단계(570e)에서 획득되는 변수 "low" 및 "high"의 값 사이의 구간은 비교적 클 것이다.

[0178] 결과적으로, 단계(570e)에서 획득되는 변수 "low" 및 "high"의 값 사이의 구간이 비교적 작으면, 다수의 구간 재규격화가 (조건 평가(570fa)의 조건의 어느 것도 충족되지 않을 만큼) "충분한" 크기로 구간을 리스케일링하도록 실행될 것이다. 따라서, 비트스트림에서의 비트의 비교적 큰 수가 변수 "value"의 정밀도를 증가시키기 위해 이용될 것이다. 이에 반해, 단계(570e)에서 획득된 구간 크기가 비교적 큰 경우, 구간 규격화(570fa 및 570fb)의 작은 수의 반복만이 변수 "low" 및 "high"의 값 사이의 구간을 "충분한" 크기로 재규격화하기 위해 필요로 될 것이다. 따라서, 비트스트림에서의 비트의 비교적 작은 수만이 변수 "value"의 정밀도를 증가시켜, 다음 심볼의 디코딩을 준비하기 위해 이용될 것이다.

[0179] 상술한 바를 요약하면, 비교적 높은 확률을 포함하고, 선택된 누적-빈도-테이블의 엔트리에 의해 큰 구간이 관련되는 심볼이 디코딩될 경우, 비트의 비교적 작은 수만이 다음 심볼의 디코딩을 허용하기 위해 비트스트림에서 판독될 것이다. 이에 반해, 비교적 적은 확률을 포함하고, 선택된 누적-빈도-테이블의 엔트리에 의해 작은 구간이 관련되는 심볼이 디코딩될 경우, 비트의 비교적 큰 수가 다음 심볼의 디코딩을 준비하기 위해 비트스트림에서 취해질 것이다.

[0180] 따라서, 누적-빈도-테이블의 엔트리는 서로 다른 심볼의 확률을 반영하고, 또한 심볼의 시퀀스를 디코딩하기 위해 필요한 비트의 수를 반영한다. 컨텍스트에 따라, 즉, 이전에 디코딩된 심볼 (또는 스펙트럼 값)에 따라 누적-빈도-테이블을 변화시킴으로써, 예컨대, 컨텍스트에 따라 서로 다른 누적-빈도-테이블을 선택함으로써, 서로 다른 심볼 사이의 확률적 의존성이 활용되어, 다음 (또는 인접한) 심볼의 특정 비트레이트 효율적인 인코딩을 허용할 수 있다.

[0181] 상술한 바를 요약하면, 도 5g를 참조로 설명된 함수 "arith_decode()"는, (복귀 변수 "symbol"에 의해 나타내는 심볼 값으로 설정될 수 있는) 최상위 비트-플레인 값 m을 결정하기 위해 함수 "arith_get_pk()"에 의해 복귀된 인덱스 "pki"에 상응하는 누적-빈도-테이블 "arith_cf_m[pki][]"이라 한다.

[0182] 6.7 이스케이프 메카니즘

[0183] (함수 "arith_decode()"에 의해 심볼 값으로서 복귀되는) 디코딩된 최상위 비트-플레인 값 m이 이스케이프 심볼 "ARITH_ESCAPE"이지만, 추가적 최상위 비트-플레인 값 m이 디코딩되고, 변수 "lev"는 1씩 증가된다. 따라서, 정보는 최상위 비트-플레인 값 m의 숫자 중요성(numeric significance) 뿐만 아니라 디코딩되는 하위 비트-플레인의 수에 관해 획득된다.

[0184] 이스케이프 심볼 "ARITH_ESCAPE"이 디코딩되면, 레벨 변수 "lev"는 1씩 증가된다. 따라서, 함수 "arith_get_pk"으로 입력되는 상태 값은 또한 최상위 비트 (비트 24 이상)에 의해 나타내는 값이 알고리즘

(312ba)의 다음 반복을 위해 증가된다는 점에서 수정된다.

[0185] 6.8 도 5h에 따른 컨텍스트 업데이트

[0186] 스펙트럼 값이 완전히 디코딩되면(즉, 최하위 비트-플레인의 모두가 추가되면), 컨텍스트 테이블 q 및 qs은 함수 "arith_update_context(a,i,lg)"를 호출하여 업데이트된다. 다음에는, 함수 "arith_update_context(a,i,lg)"에 관한 상세 사항이 도 5h를 참조로 설명되며, 도 5h는 상기 함수의 의사 프로그램 코드 표현을 도시한다.

[0187] 함수 "arith_update_context()"는, 입력 변수로서, 디코딩 양자화된 스펙트럼 계수 a, 디코딩되는 스펙트럼 값(또는 디코딩된 스펙트럼 값)의 인덱스 i, 및 현재 오디오 프레임과 관련된 스펙트럼 값(또는 계수)의 수 lg를 받는다.

[0188] 단계(580)에서, 현재 디코딩 양자화된 스펙트럼 값(또는 계수) a는 컨텍스트 테이블 또는 컨텍스트 어레이 q로 복사된다. 따라서, 컨텍스트 테이블 q의 엔트리 q[1][i]는 a로 설정된다. 또한 변수 "a0"는 "a"의 값으로 설정된다.

[0189] 단계(582)에서, 컨텍스트 테이블 q의 레벨 값 q[1][i].1이 결정된다. 디폴트에 의해, 컨텍스트 테이블 q의 레벨 값 q[1][i].1은 0으로 설정된다. 그러나, 현재 코딩된 스펙트럼 값 a의 절대값이 4보다 큰 경우, 레벨 값 q[1][i].1은 증가된다.

[0190] 각 증가로, 변수 "a"는 1 비트씩 오른쪽으로 시프트된다. 레벨 값 q[1][i].1의 증가는 변수 a0의 절대값이 4 이하일 때까지 반복된다.

[0191] 단계(584)에서, 컨텍스트 테이블 q의 2-비트 컨텍스트 값 q[1][i].c이 설정된다. 2-비트 컨텍스트 값 q[1][i].c은 현지 디코딩된 스펙트럼 값 a이 0과 동일할 경우에 0의 값으로 설정된다. 그렇지 않으면, 디코딩된 스펙트럼 값 a의 절대값이 1 이하이면, 2-비트 컨텍스트 값 q[1][i].c은 1로 설정된다. 그렇지 않으면, 현재 디코딩된 스펙트럼 값 a의 절대값이 3 이하이면, 2-비트 컨텍스트 값 q[1][i].c은 2로 설정된다. 그렇지 않으면, 즉, 현재 디코딩된 스펙트럼 값 a의 절대값이 3보다 크면, 2-비트 컨텍스트 값 q[1][i].c은 3으로 설정된다. 따라서, 2-비트 컨텍스트 값 q[1][i].c은 현재 디코딩된 스펙트럼 값 a의 매우 거친(coarse) 양자화에 의해 획득된다.

[0192] 현재 디코딩된 스펙트럼 값의 인덱스 i가 프레임의 계수(스펙트럼 값)의 수 lg와 동일하고, 즉, 프레임의 마지막 스펙트럼 값이 디코딩되었으며, 코어 모드가 ("core_mode==1"에 의해 나타내는) 선형-예측-도메인 코어 모드인 경우에만 수행되는 다음 단계(586)에서, 엔트리 q[1][j].c는 컨텍스트 테이블 qs[k]로 복사된다. 참조번호(586)에 도시된 바와 같이 복사가 수행되어, 현재 프레임의 스펙트럼 값의 수 lg가 엔트리 q[1][j].c를 컨텍스트 테이블 qs[k]로 복사하기 위해 고려되도록 한다. 게다가, 변수 "previous_lg"는 값 1024을 취한다.

[0193] 그러나, 대안적으로, 컨텍스트 테이블 q의 엔트리 q[1][j].c는 현재 디코딩된 스펙트럼 계수의 인덱스 i가 lg의 값에 도달하고, 코어 모드가 ("core_mode==0"에 의해 나타내는) 주파수-도메인 코어 모드인 경우에 컨텍스트 테이블 qs[j]로 복사된다.

[0194] 이 경우에, 변수 "previous_lg"는 1024 값과 프레임의 스펙트럼 값의 수 lg 사이의 최소값으로 설정된다.

[0195] 6.9 디코딩 프로세스의 요약

[0196] 다음에는, 디코딩 프로세스가 간략하게 요약된다. 상세 사항에 대해서는, 상술한 논의 및 또한 도 3, 4 및 5a 내지 5i에 대한 참조가 행해진다.

[0197] 양자화된 스펙트럼 계수 a는 잡음없이 복사되어, 전송되며, 최저 주파수 계수로부터 시작하여, 최고 주파수 계수로 진행한다.

[0198] 고급-오디오 코딩(AAC)의 계수는 어레이 "x_ac_quant[g][win][sfb][bin]"에 저장되고, 잡음없는 코딩 코드워드의 전송의 순서는, 이들이 어레이에 수신되고 저장된 순서로 디코딩될 때, bin은 가장 빠르게 증가하는 인덱스이며, g는 가장 느리게 증가하는 인덱스이도록 한다. 인덱스 bin는 주파수 bin를 명시한다. 인덱스 "sfb"는 스케일 팩터 대역을 명시한다. 인덱스 "win"는 윈도우를 명시한다. 인덱스 "g"는 오디오 프레임을 명시한다.

[0199] 변환-코딩된-여기에서의 계수는 어레이 "x_tcx_invquant[win][bin]"에 직접 저장되고, 잡음없는 코딩 코드워드의 전송의 순서는, 이들이 어레이에 수신되고 저장된 순서로 디코딩될 때, bin은 가장 빠르게 증가하는 인

덱스이며, "win"는 가장 느리게 증가하는 인덱스이도록 한다.

[0200] 첫째로, 컨텍스트 테이블 또는 어레이 "qs"에 저장되는 세이빙된(saved) 과거 컨텍스트와 (컨텍스트 테이블 또는 어레이 q에 저장되는) 현재 프레임 q의 컨텍스트 사이에서 맵핑이 행해진다. 과거 컨텍스트 "qs"는 주파수 라인마다 (또는 주파수 bin마다) 2-비트로 저장된다.

[0201] 컨텍스트 테이블 "qs"에 저장되는 세이빙된 과거 컨텍스트와 컨텍스트 테이블 "q"에 저장되는 현재 프레임의 컨텍스트 사이의 맵핑은 함수 "arith_map_context()"를 이용하여 수행되며, 이의 의사-프로그램-코드 표현은 도 5a에 도시된다.

[0202] 잡음없는 디코더는 부호화 양자화된 스펙트럼 계수 "a"를 출력한다.

[0203] 처음에, 컨텍스트의 상태는 디코딩하는 양자화된 스펙트럼 계수 주위의(surrounding) 이전에 디코딩된 스펙트럼 계수에 기초하여 계산된다. 컨텍스트 s의 상태는 함수 "arith_get_context()"에 의해 복귀되는 값의 24의 제 1 비트에 상응한다. 복귀되는 값의 24 비트 이외의 비트는 예측된 비트-플레인-레벨 lev0에 상응한다. 변수 "lev"는 lev0로 초기화된다. 함수 "arith_get_context"의 의사 프로그램 코드 표현은 도 5b 및 5c에 도시된다.

[0204] 상태 s 및 예측된 레벨 "lev0"이 알려지면, 최상위 2-비트 와이즈(wise) 플레인 m은 컨텍스트 상태에 상응하는 확률 모델에 상응하는 적절한 누적-빈도-테이블로 공급되는 함수 "arith_decode()"를 이용하여 디코딩된다.

[0205] 상응(correspondence)은 함수 "arith_get_pk()"에 의해 행해진다.

[0206] 함수 "arith_get_pk()"의 의사-프로그램-코드 표현은 도 5e에 도시된다.

[0207] 함수 "arith_get_pk()"를 대신할 수 있는 다른 함수 "get_pk"의 의사 프로그램 코드는 도 5f에 도시된다. 함수 "arith_get_pk()"를 대신할 수 있는 다른 함수 "get_pk"의 의사 프로그램 코드는 도 5d에 도시된다.

[0208] 누적-빈도-테이블 "arith_cf_m[pki][]"로 불리는 함수 "arith_decode()"를 이용하여 값 m은 디코딩되며, 여기서, "pki"는 함수 "arith_get_pk()"(또는, 대안적으로, 함수 "get_pk()")에 의해 복귀되는 인덱스에 상응한다.

[0209] 산술 코더는 스케일링에 의한 태그 생성 방법을 이용하는 정수 구현(integer implementation)이다(예컨대, K. Sayood "Introduction to Data Compression" third edition, 2006, Elsevier Inc. 참조). 도 5g에 도시된 의사-C-코드는 이용되는 알고리즘을 나타낸다.

[0210] 디코딩된 값 m이 이스케이프 심볼 "ARITH_ESCAPE"이면, 다른 값 m은 디코딩되고, 변수 "lev"는 1 씩 증가된다. 값 m이 이스케이프 심볼 "ARITH_ESCAPE"이 아니면, 나머지 비트-플레인은 누적-빈도-테이블 "arith_cf_r[]"로 함수 "arith_decode()"를 "lev" 번 호출하여 최상위 레벨에서 최하위 레벨까지 디코딩된다. 상기 누적-빈도-테이블 "arith_cf_r[]"은, 예컨대, 고른(even) 확률 분포를 나타낼 수 있다.

[0211] 디코딩된 비트 플레인 r은 다음과 같은 방식으로 이전에 디코딩된 값 m의 리파인닝(refining)을 허용한다:

[0212] a = m;

[0213] for (i=0; i<lev;i++) {

[0214] r = arith_decode (arith_cf_r,2);

[0215] a = (a<<1) | (r&1);

[0216] }

[0217] 스펙트럼 양자화된 계수가 완전히 디코딩되면, 컨텍스트 테이블 q 또는 저장된 컨텍스트 qs는 디코딩할 다음 양자화된 스펙트럼 계수에 대해 함수 "arith_update_context()"에 의해 업데이트된다.

[0218] 함수 "arith_update_context()"의 의사 프로그램 코드 표현은 도 5h에 도시된다.

[0219] 게다가, 정의의 레전드는 도 5i에 도시된다.

[0220] 7. 맵핑 테이블

[0221] 본 발명에 따른 실시예에서, 특히 유리한 테이블 "ari_s_hash" 및 "ari_gs_hash" 및 "ari_cf_m"은, 도 5d와

관련하여 논의되었던 함수 "get_pk"의 실행, 또는 도 5e와 관련하여 논의되었던 함수 "arith_get_pk"의 실행, 또는 도 5f와 관련하여 논의된 함수 "get_pk"의 실행, 및 도 5g와 관련하여 논의된 함수 "arith_decode"의 실행을 위해 이용된다.

[0222] 7.1. 도 17에 따른 테이블 "ari_s_hash[387]"

[0223] 도 5d와 관련하여 논의된 함수 "get_pk"에 의해 이용되는 테이블 "ari_s_hash"의 특히 유리한 구현의 콘텐츠는 도 17의 테이블에 도시된다. 또한, 도 17의 테이블은 테이블 "ari_s_hash[387]"의 387 엔트리를 열거하는 것으로 언급되어야 한다. 또한, 도 17의 테이블 표현은, 제 1 값 "0x00000200"이 요소 인덱스(또는 테이블 인덱스) 0을 가진 테이블 엔트리 "ari_s_hash[0]"에 상응하고, 마지막 값 "0x03D0713D"이 요소 인덱스 또는 테이블 인덱스 386을 가진 테이블 엔트리 "ari_s_hash[386]"에 상응하도록 요소 인덱스의 순서로 요소를 도시하는 것으로 언급되어야 한다. "0x"는 테이블 "ari_s_hash"의 테이블 엔트리가 16 진수 포맷으로 표시됨을 나타내는 것으로 더 언급되어야 한다. 더욱이, 도 17에 따른 테이블 "ari_s_hash"의 테이블 엔트리는 함수 "get_pk"의 제 1 테이블 평가(540)의 실행을 허용하기 위해 숫자 순서로 배치된다.

[0224] 테이블 "ari_s_hash"의 테이블 엔트리의 최상위 24 비트는 상태 값을 나타내지만, 최하위 8 비트는 맵핑 규칙 인덱스 값 pki을 나타내는 것으로 더 언급되어야 한다.

[0225] 따라서, 테이블 "ari_s_hash"의 엔트리는 맵핑 규칙 인덱스 값 "pki"로의 상태 값의 "다이렉트 히트" 맵핑을 나타낸다.

[0226] 7.2 도 18에 따른 테이블 "ari_gs_hash"

[0227] 테이블 "ari_gs_hash"의 특히 유리한 실시예의 콘텐츠는 도 18의 테이블에 도시된다. 여기서,테이블 18의 테이블은 테이블 "ari_gs_hash"의 엔트리를 열거하는 것으로 언급되어야 한다. 상기 엔트리는, 예컨대, "i"로 명시되는 1차원 정수 타입 엔트리 인덱스(또한 "요소 인덱스" 또는 "어레이 인덱스" 또는 "테이블 인덱스"로 명시됨)에 의해 참조된다. 전체 225 엔트리를 포함하는 테이블 "ari_gs_hash"은 도 5d에 나타낸 함수 "get_pk"의 제 2 테이블 평가(544)에 의한 이용에 적합한 것으로 언급되어야 한다.

[0228] 테이블 "ari_gs_hash"의 엔트리는 0과 224 사이의 테이블 인덱스 값 i에 대한 테이블 인덱스 i의 오름차순 순서로 열거되는 것으로 언급되어야 한다. 용어 "0x"는 테이블 엔트리가 16 진수 포맷으로 설명되는 것을 나타낸다. 따라서, 제 1 테이블 엔트리 "0X00000401"는 테이블 인덱스 0을 가진 테이블 엔트리 "ari_gs_hash[0]"에 상응하고, 마지막 테이블 엔트리 "0Xffffff3f"는 테이블 인덱스 224를 가진 테이블 엔트리 "ari_gs_hash[224]"에 상응한다.

[0229] 또한, 테이블 엔트리는 테이블 엔트리가 함수 "get_pk"의 제 2 테이블 평가(544)에 적합하도록 숫자상 오름차순 방식으로 순서가 이루어지는 것으로 언급되어야 한다. 테이블 "ari_gs_hash"의 테이블 엔트리의 최상위 24 비트는 상태 값의 범위 사이의 경계를 나타내고, 엔트리의 8 최하위 비트는 24 최상위 비트에 의해 정의되는 상태 값의 범위와 관련된 맵핑 규칙 인덱스 값 "pki"을 나타낸다.

[0230] 7.3 도 19에 따른 테이블 "ari_cf_m"

[0231] 도 19는 64 누적-빈도-테이블 "ari_cf_m[pki][9]"의 세트를 도시하며, 이 중 하나는, 예컨대, 함수 "arith_decode"의 실행을 위해, 즉, 최상위 비트-플레인 값의 디코딩을 위해 오디오 인코더(100, 700) 또는 오디오 디코더(200, 800)에 의해 선택된다. 도 19에 도시된 64 누적-빈도-테이블 중 선택된 것은 함수 "arith_decode()"의 실행에서 테이블 "cum_freq[]"의 함수를 취한다.

[0232] 도 10에서 볼 수 있듯이. 각 라인은 9 엔트리를 갖는 누적-빈도-테이블을 나타낸다. 예컨대, 제 1 라인(1910)은 "pki=0"에 대한 누적-빈도-테이블의 9 엔트리를 나타낸다. 제 2 라인(1912)은 "pki=1"에 대한 누적-빈도-테이블의 9 엔트리를 나타낸다. 최종으로, 제 64 라인(1964)은 "pki=63"에 대한 누적-빈도-테이블의 9 엔트리를 나타낸다. 따라서, 도 19는 "pki=0" 내지 "pki=63"에 대한 64의 서로 다른 누적-빈도-테이블을 효과적으로 나타내며, 64 누적-빈도-테이블의 각각은 단일 라인에 의해 나타내고, 상기 누적-빈도-테이블의 각각은 9 엔트리를 포함한다.

[0233] 라인(예컨대, 라인(1910) 또는 라인(1912) 또는 라인(1964)) 내에서, 가장 왼쪽 값은 누적-빈도-테이블의 제 1 엔트리를 나타내고, 가장 오른쪽 값은 누적-빈도-테이블의 마지막 엔트리를 나타낸다.

[0234] 따라서, 도 19의 테이블 표현의 각 라인(1910, 1912, 1964)은 도 5g에 따른 함수 "arith_decode"에 의한 이용을 위한 누적-빈도-테이블의 엔트리를 나타낸다. 함수 "arith_decode"의 입력 변수 "cum_freq[]"는 테이블

"ari_cf_m"의 (9 엔트리의 개별 라인으로 나타내는) 64 누적-빈도-테이블 중 어떤 것이 현재 스펙트럼 계수의 디코딩을 위해 이용되어야 하는지를 나타낸다.

[0235] 7.4 도 20에 따른 테이블 "ari_s_hash"

[0236] 도 20은 도 5e 또는 5f에 따른 대안적 함수 "arith_get_pk()" 또는 "get_pk()"와 함께 이용될 수 있는 테이블 "ari_s_hash"에 대한 대안을 도시한다.

[0237] 도 20에 따른 테이블 "ari_s_hash"은 테이블 인덱스의 오름차순 순서로 도 20에 열거되는 386 엔트리를 포함한다. 따라서, 제 1 테이블 값 "0x0090D52E"는 테이블 인덱스 0을 가진 테이블 엔트리 "ari_s_hash[0]"에 상응하고, 마지막 테이블 엔트리 "0x03D0513C"는 테이블 인덱스 386을 가진 테이블 엔트리 "ari_s_hash[386]"에 상응한다.

[0238] "0x"는 테이블 엔트리가 16 진수 포맷으로 표시되는 것을 나타낸다. 테이블 "ari_s_hash"의 엔트리의 24 최상위 비트는 유효 상태(significant states)를 나타내고, 테이블 "ari_s_hash"의 엔트리의 8 최하위 비트는 맵핑 규칙 인덱스 값을 나타낸다.

[0239] 따라서 테이블 "ari_s_hash"의 엔트리는 맵핑 규칙 인덱스 값 "pki"으로의 유효 상태의 맵핑을 나타낸다.

[0240] 8. 성능 평가 및 이점

[0241] 본 발명에 따른 실시예들은 상술한 바와 같이 업데이트된 함수 (또는 알고리즘) 및 테이블의 업데이트된 세트를 이용하여, 계산 복잡도, 메모리 요구 사항 및 코딩 효율 사이의 개선된 트레이드오프를 획득한다.

[0242] 일반적으로, 본 발명에 따른 실시예들은 향상된 스펙트럼 잡음없는 코딩을 생성한다.

[0243] 본 설명은 스펙트럼 계수의 향상된 스펙트럼 소리가 잡음없는 코딩에 관한 CE에 대한 실시예들을 설명한다. 제안된 기법은 USAC 초안 표준의 작업 초안 4에서 설명된 바와 같이 "원래의" 컨텍스트 기반 산술 코딩 기법에 기초하지만, 잡음없는 코딩 성능을 유지하면서 메모리 요구 사항(RAM, ROM)을 상당히 줄인다. WD3 (즉, USAC 초안 표준 작업 초안 3에 따라 비트스트림을 제공하는 오디오 인코더의 출력)의 무손실 트랜스코딩은 가능한 것으로 입증되었다. 여기에 설명된 기법은, 일반적으로, 확장 가능하고, 메모리 요구 사항 및 인코딩 성능 사이의 추가 대안적 트레이드오프를 가능하게 한다. 본 발명에 따른 실시예들은 USAC 초안 표준의 작업 초안 4에서 이용되는 바와 같은 스펙트럼 잡음없는 코딩 기법을 대체하는데 도움을 준다.

[0244] 여기에 설명된 산술 코딩 기법은 참조 모델 0(RM0) 또는 USAC 초안 표준의 작업 초안 4(WD4)에서와 같은 기법에 기초한다. 이전에 주파수 또는 시간 모델에서의 스펙트럼 계수는 컨텍스트이다. 이러한 컨텍스트는 산술 코더 (인코더 또는 디코더)에 대한 누적-빈도-테이블의 선택에 이용된다. WD4에 따른 실시예에 비해, 컨텍스트 모델링은 더욱 향상되고, 심볼 확률을 보유한 테이블은 리트레인(retrain)되었다. 서로 다른 확률 모델의 수는 32에서 64로 증가되었다.

[0245] 본 발명에 따른 실시예들은 테이블 크기(데이터 ROM 수요(demand))를 길이 32-비트 또는 3600 바이트의 900 워드로 줄인다. 이에 반해, USAC 초안 표준의 WD4에 따른 실시예들은 16894.5 워드 또는 76578 바이트를 필요로 한다. 정적 RAM 수요는, 본 발명에 따른 일부 실시예들에서, 코어 코더 채널당 666 워드(2664 바이트)에서 72 (288 바이트)로 감소된다. 동시에, 그것은 코딩 성능을 완벽하게 보존하고, 모든 9 동작 포인트를 통해 전체 데이터 속도에 비해 대략 1.04 % 내지 1.39 %의 이득에 도달할 수도 있다. 모든 작업 초안 3 (WD3) 비트스트림은 비트 저장소 제약(bit reservoir constraints)에 영향을 미치지 않고 무손실 방식으로 트랜스코딩될 수 있다.

[0246] 본 발명의 실시예들에 따른 제안된 기법은 확장 가능하다: 메모리 수요와 코딩 성능 사이의 유연한 트레이드오프가 가능하다. 코딩 이득에 대한 테이블 크기를 증대시킴으로써 더욱 증대될 수 있다.

[0247] 다음에는, USAC 초안 표준의 WD4에 따른 코딩 개념에 대한 간단한 논의가 여기에 설명된 개념의 이점에 대한 이해를 용이하게 하기 위해 제공될 것이다. USAC WD4에서, 컨텍스트 기반 산술 코딩 기법은 양자화된 스펙트럼 계수의 잡음없는 코딩에 이용된다. 컨텍스트로서, 이전에 주파수 또는 시간에서의 디코딩된 스펙트럼 계수가 이용된다. WD4에 따르면, 16의 스펙트럼 계수의 최대 수는 컨텍스트로서 이용되며, 그 중 12개는 이전에 시간에 있다. 컨텍스트에 이용되고, 디코딩되는 양방의 스펙트럼 계수는 4-튜플(tuples)(즉, 주파수에 이웃된 4 스펙트럼 계수, 도 10a 참조)로서 그룹화된다. 컨텍스트는 감소되어, 스펙트럼 계수의 다음 4-튜플을 디코딩하는데 사용되는 누적-빈도-테이블에 맵핑된다.

[0248] 완전한 WD4 잡음없는 코딩 기법의 경우, 16894.5 워드 (67578 바이트)의 메모리 수요(ROM)가 필요로 된다. 부가적으로, 코어-코더 채널 당 정적 ROM의 666 워드(2664 바이트)는 다음 프레임에 대한 상태를 저장하는데 필요로 된다.

[0249] 도 11a의 테이블 표현은 USAC WD4 산술 코딩 기법에 이용되는 테이블을 나타낸다.

[0250] 완전한 USAC WD4 디코더의 전체 메모리 수요는 프로그램 코드 없는 데이터 ROM에 대한 37000 워드(148000 바이트) 및 정적 RAM에 대한 10000 내지 17000 워드인 것으로 추정된다. 잡음없는 코더 테이블은 전체 데이터 ROM 수요의 대략 45%를 소비하는 것을 명확히 알 수 있다. 가장 큰 개별 테이블은 이미 4096 워드(16384 바이트)를 소비한다.

[0251] 모든 테이블 및 큰 개별 테이블의 조합의 크기의 양방은 8-32 kByte의 전형적인 범위(예컨대, ARM9e, TIC64xx 등) 내에 있는 저예산(low-budget) 휴대용 장치에 대한 고정 소수점 칩(fixed point chips)에 의해 제공되는 바와 같은 전형적인 캐시 크기를 초과하는 것으로 발견되었다. 이것은 테이블의 세트가 아마 데이터에 대한 빠른 랜덤 액세스를 가능하게 하는 빠른 데이터 RAM에 저장될 수 없다는 것을 의미한다. 이것은 전체 디코딩 프로세스를 느리게 한다.

[0252] 다음에는, 제안된 새로운 기법이 간략하게 설명될 것이다.

[0253] 상술한 문제점을 극복하기 위해, 향상된 잡음없는 코딩 기법이 USAC 초안 표준의 WD4에서와 같이 기법을 대체하도록 제안된다. 컨텍스트 기반 산술 코딩 기법으로서, 그것은 USAC 초안 표준의 WD4의 기법에 기초하지만, 컨텍스트에서 누적-빈도-테이블의 유도를 위한 수정된 기법을 특징으로 한다. 더욱이, 컨텍스트 유도 및 심볼 코딩은 (USAC 초안 표준의 WD4에서와 같이 4-튜플에 반대되는) 단일 스펙트럼 계수의 입도(granularity)에서 수행된다. 전체적으로, 7 스펙트럼 계수가 컨텍스트(적어도 일부의 경우)에 이용된다.

[0254] 맵핑에서 감소시킴으로써, 전체 64 확률 모델 또는 누적 빈도 테이블(WD4: 32에서)에서 하나가 선택된다.

[0255] 도 10b는 (제로 영역 검출에 이용되는 컨텍스트가 도 10b에 도시되지 않은) 제안된 기법에 이용되는 바와 같이, 상태 계산을 위한 컨텍스트의 그래픽 표현을 도시한다.

[0256] 다음에는, 제안된 코딩 기법을 이용하여 달성될 수 있는 메모리 수요의 감소에 관한 간략한 논의가 제공될 것이다. 제안된 새로운 기법은 900 워드(3600 바이트)의 전체 ROM 수요를 나타낸다(제안된 코딩 기법에서 이용되는 바와 같은 테이블을 나타내는 도 11b의 테이블 참조).

[0257] USAC 초안 표준의 WD4에서 잡음없는 코딩 기법의 ROM 수요에 비해, ROM 수요는 15994.5 워드(64978 바이트)만큼 감소된다(또한, 제안된 바와 같은 잡음없는 코딩 기법 및, USAC 초안 표준의 WD4에서의 잡음없는 코딩 기법의 그래픽 표현을 도시하는 도 12a 참조). 이것은 완전한 USAC 디코더의 전체 ROM 수요를 대략 37000 워드에서 대략 21000 워드까지, 또는 43% 이상 감소시킨다(USAC 초안 표준의 WD4에 따를 뿐만 아니라 현재 제안에 따른 전체 USAC 디코더 데이터 ROM 수요의 그래픽 표현을 도시하는 도 12b 참조).

[0258] 더욱이, 다음 프레임(정적 RAM)에서 컨텍스트 유도에 필요한 정보의 양은 또한 감소된다. WD4에 따르면, 해상도 10-비트의 4-튜플 당 그룹 인덱스에 추가하는 전형적으로 16-비트의 해상도를 가진 계수의 완전한 세트(최대 1152)는 합계 코어-코더 채널 당 666 워드(2664 바이트)(완전한 USAC WD4 디코더: 대략 10000 내지 17000 워드)까지 저장될 필요가 있다.

[0259] 본 발명에 따른 실시예들에 이용되는 새로운 기법은 영구적인 정보를 스펙트럼 계수 당 2-비트만으로 줄여, 코어-코더 채널 당 전체적으로 합계 72 워드(288 바이트)로 된다. 정적 메모리에 대한 수요는 594 워드(2376 바이트)만큼 줄일 수 있다.

[0260] 다음에는, 코딩 효율의 가능한 증가에 관한 몇 가지 상세 사항이 설명된다. 새로운 제안에 따른 실시예들의 코딩 효율은 USAC 초안 표준의 WD3에 따라 참조 품질 비트스트림에 대해 비교되었다. 이러한 비교는 참조 소프트웨어 디코더에 기초하여 트랜스코더에 의해 수행되었다. USAC의 초안 표준 및 제안된 코딩 기법의 WD3에 따른 잡음없는 코딩의 비교에 관한 상세 사항에 대해, 테스트 배치의 개략적 표현을 도시하는 도 9에 대한 참조가 행해진다.

[0261] WD3 또는 USAC 초안 표준의 WD4에 따라 실시예들에 비해 메모리 수요가 본 발명에 따른 실시예들에서 크게 감소될지라도, 코딩 효율은 유지될 뿐만 아니라, 약간 증가된다. 코딩 효율은 평균하여 1.04 % 내지 1.39 %만큼 증가된다. 상세 사항에 대해, 본 발명의 실시예에 따른 작업 초안 산술 코더 및 오디오 코더(예컨대, USAC 오디오 코더)를 이용하는 USAC 코더에 의해 생성되는 평균 비트레이트의 테이블 표현을 도시하는 도 13a의 테이

블에 대한 참조가 행해진다.

[0262] 비트 저장소 채움(fill) 레벨을 측정함으로써, 제안된 잡음없는 코딩은 모든 동작 포인트에 대한 WD3 비트스트림을 무손실로 트랜스코딩할 수 있는 것으로 나타났다. 상세 사항에 대해, USAC WD3에 따른 오디오 코더 및 본 발명의 실시예에 따른 오디오 코더에 대한 비트 저장소 제어의 테이블 표현을 도시하는 도 13b의 테이블에 대한 참조가 행해진다

[0263] 동작 모드 마다 평균 비트레이트, 프레임 단위의(on a frame basis) 최소, 최대 및 평균 비트레이트 및 프레임 단위의 최상/최악의 경우의 성능은 도 14, 15 및 16의 테이블에서 발견될 수 있으며, 도 14의 테이블은 USAC WD3에 따른 오디오 코더 및 본 발명의 실시예에 따른 오디오 코더에 대한 평균 비트레이트의 테이블 표현을 도시하고, 도 15의 테이블은 프레임 단위의 USAC 오디오 코더의 최소, 최대 및 평균 비트레이트의 테이블 표현을 도시하며, 도 16의 테이블은 프레임 단위의 최상 및 최악의 경우의 테이블 표현을 도시한다.

[0264] 게다가, 본 발명에 따른 실시예들은 양호한 확장성을 제공하는 것으로 언급되어야 한다. 테이블 크기를 적응시킴으로써, 메모리 요구 사항, 계산 복잡도 및 코딩 효율 사이의 트레이드오프가 이러한 요구 사항에 따라 조정될 수 있다.

[0265] 9. 비트스트림 구문(syntax)

[0266] 9-1. 스펙트럼 잡음없는 코더의 페이로드

[0267] 다음에는, 스펙트럼 잡음없는 코더의 페이로드에 관한 몇 가지 상세 사항이 설명될 것이다. 일부 실시예들에서, 예컨대, 소위 선형-예측-도메인, "코딩 모드" 및 "주파수-도메인" 코딩 모드와 같은 다수의 서로 다른 코딩 모드가 있다. 선형-예측-도메인 코딩 모드에서, 잡음 형상화(noise shaping)는 오디오 신호의 선형-예측 분석에 기초하여 수행되고, 잡음-형상화된 신호는 주파수-도메인으로 인코딩된다. 주파수-도메인 모드에서, 잡음 형상화는 심리 음향학 분석에 기초하여 수행되고, 오디오 콘텐츠의 잡음-형상화된 버전은 주파수-도메인으로 인코딩된다.

[0268] "선형-예측 도메인" 코딩된 신호 및 "주파수-도메인" 코딩된 신호의 양방으로부터의 스펙트럼 계수는 스칼라 양자화되어, 적응 컨텍스트 의존 산술 코딩에 ㅇ의해 잡음없이 코딩된다. 양자화된 계수는 최저 주파수에서 최고 주파수로 전송된다. 각 개별 양자화된 계수는 최상위 2-비트-와이즈 플레인 m 및 나머지 하위 비트-플레인 r로 분할된다. 값 m은 계수의 이웃에 따라 코딩된다. 나머지 하위 비트-플레인 r은 컨텍스트를 고려하지 않고 엔트로피-인코딩된다. 값 m 및 r은 산술 코더의 심볼을 형성한다.

[0269] 상세한 산술 디코딩 절차는 여기에 설명되어 있다.

[0270] 9.2. 구문 요소

[0271] 다음에는, 산술적으로-인코딩된 스펙트럼 정보를 반송하는 비트스트림의 비트스트림 구문은 도 6a 내지 6h와 관련하여 설명될 것이다.

[0272] 도 6a는 소위 USAC 원시(raw) 데이터 블록("usac_raw_data_block()")의 구문 표현을 도시한 것이다.

[0273] USAC 원시 데이터 블록은 하나 이상의 단일 채널 요소("single_channel_element()") 및/또는 하나 이상의 채널 쌍 요소("channel_pair_element()")를 포함한다.

[0274] 이제 도 6b를 참조하면, 단일 채널 요소의 구문이 설명된다. 단일 채널 요소는 코어 모드에 따라 선형-예측-도메인 채널 스트림("lpd_channel_stream()") 또는 주파수-도메인 채널 스트림("fd_channel_stream()")을 포함한다.

[0275] 도 6c는 채널 쌍 요소의 구문 표현을 도시한다. 채널 쌍 요소는 코어 모드 정보("core_mode0", "core_mode1")를 포함한다. 게다가, 채널 쌍 요소는 구성 정보 "ics_info()"를 포함할 수 있다. 부가적으로, 코어 모드 정보에 따라, 채널 쌍 요소는 제 1 채널과 관련된 선형-예측-도메인 채널 스트림 또는 주파수-도메인 채널 스트림을 포함하고, 채널 쌍 요소는 또한 제 2 채널과 관련된 선형-예측-도메인 채널 스트림 또는 주파수-도메인 채널 스트림을 포함한다.

[0276] 구문 표현이 도 6d에 도시된 구성 정보 "ics_info()"는 본 발명에 대한 특정 관련성이 없는 다수의 서로 다른 구성 정보 항목을 포함한다.

[0277] 구문 표현이 도 6e에 도시된 주파수-도메인 채널 스트림("fd_channel_stream ()")은 이득 정보("global_gain") 및 구성 정보("ics_info ()")를 포함한다. 게다가, 주파수-도메인 채널 스트림은, 서로 다른

스케일 팩터 대역의 스펙트럼 값의 스케일링에 이용되는 스케일 팩터를 나타내고, 예컨대, 스케일러(150) 및 리스케일러(rescaler)(240)에 의해 적용되는 스케일 팩터 데이터("scale_factor_data ()")를 포함한다. 주파수-도메인 채널 스트림은 또한 산술적으로-인코딩된 스펙트럼 값을 나타내는 산술적으로-코딩된 스펙트럼 데이터("ac_spectral_data ()")를 포함한다.

[0278] 구문 표현이 도 6f에 도시된 산술적으로-코딩된 스펙트럼 데이터("ac_spectral_data()")는 상술한 바와 같이 선택적으로 컨텍스트를 재설정하는 데 이용되는 선택적 산술 재설정 플래그("arith_reset_flag")를 포함한다. 게다가, 산술적으로-코딩된 스펙트럼 데이터는 산술적으로-코딩된 스펙트럼 값을 반송하는 다수의 산술-데이터 블록("arith_data")을 포함한다. 다음에 논의되는 바와 같이, 산술적으로-코딩된 데이터 블록의 구조는 (변수 "num_bands"로 나타내는) 주파수 대역의 수 및 또한 산술 재설정 플래그의 상태에 의존한다.

[0279] 산술적으로-인코딩된 데이터 블록의 구조는 상기 산술적으로-코딩된 데이터 블록의 구문 표현을 도시한 도 6g과 관련하여 설명될 것이다. 산술적으로-코딩된 데이터 블록 내의 데이터 표현은 인코딩되는 스펙트럼 값의 수 lg, 산술 재설정 플래그의 상태 및 또한 컨텍스트, 즉 이전에 인코딩 스펙트럼 값에 의존한다.

[0280] 스펙트럼 값의 현재 세크의 인코딩에 대한 컨텍스트는 참조 번호(660)에 도시된 컨텍스트 결정 알고리즘에 따라 결정된다. 컨텍스트 결정 알고리즘에 대한 상세 사항은 도 5a와 관련하여 상술되었다. 산술적으로-인코딩된 데이터 블록은 코드워드의 각 세트가 스펙트럼 값을 나타내는 코드워드의 lg 세트를 포함한다. 코드워드의 세트는 1 비트와 20 비트 사이에서 이용하는 스펙트럼 값의 최상위 비트-플레인 값 m을 나타내는 산술 코드워드 "acod_m [pki][m]"를 포함한다. 게다가, 코드워드의 세트는 스펙트럼 값이 정확한 표현을 위해 최상위 비트 플레인보다 더 많은 비트 플레인을 필요로 할 경우에 하나 이상의 코드워드 "acod_r[r]"를 포함한다. 코드워드 "acod_r[r]"는 1 비트와 20 비트 사이에서 이용하는 하위 비트-플레인을 나타낸다.

[0281] 그러나, 하나 이상의 하위 비트-플레인은 (최상위 비트 플레인 이외에) 스펙트럼 값의 적절한 표현을 위해 필요로 되며, 이것은 하나 이상의 산술 이스케이프 코드워드("ARITH_ESCAPE")에 의해 신호화된다. 따라서, 일반적으로, 스펙트럼 값에 대해, 얼마나 많은 비트 플레인(최상위 비트 플레인 및 아마도 하나 이상의 추가적 하위 비트 플레인)이 필요로 되는 지가 판단된다고 할 수 있다. 하나 이상의 하위 비트 플레인이 필요로 되면, 이것은, 누적-빈도-테이블-인덱스가 변수 pki에 의해 주어지는 현재-선택된 누적-빈도-테이블에 따라 인코딩되는 하나 이상의 산술 이스케이프 코드워드 "acod_m [pki][ARITH_ESCAPE]"에 의해 신호화된다. 게다가, 하나 이상의 산술 이스케이프 코드워드가 비트스트림에 포함되는 경우에, 컨텍스트는 참조 번호(664, 662)에서 알 수 있듯이 적응된다. 하나 이상의 산술 이스케이프 코드워드에 뒤따라, 산술 코드워드 "acod_m [pki][m]"는 참조 번호(663)에 도시된 바와 같이 비트스트림에 포함되며, 여기서, pki는 (산술 이스케이프 코드워드의 포함에 의해 유발되는 컨텍스트 적응을 고려하는) 현재-유효한 확률 모델 인덱스를 명시하며, m은 인코딩 또는 디코딩되는 스펙트럼 값의 최상위 비트-플레인 값을 명시한다.

[0282] 상술한 바와 같이, 어떤 하위-비트 플레인의 존재는, 각각 최하위 비트 플레인의 하나의 비트를 나타내는 하나 이상의 코드워드 "acod_r[r]"의 존재를 생성한다. 하나 이상의 코드워드 "acod_r[r]"는 일정하고, 컨텍스트에 무관한 상응하는 누적-빈도-테이블에 따라 인코딩된다.

[0283] 게다가, 컨텍스트는, 참조 번호(668)에 도시된 바와 같이, 컨텍스트가 통상적으로 2개의 후속 스펙트럼 값의 인코딩에 대해 서로 다르도록 각 스펙트럼 값의 인코딩 후에 업데이트되는 것으로 언급되어야 한다.

[0284] 도 6h는 산술적으로-인코딩된 데이터 블록의 구문을 정의하는 정의 및 헬프 요소의 레전드를 도시한 것이다.

[0285] 상술한 바를 요약하면, 오디오 코더(100)에 의해 제공될 수 있고, 오디오 디코더(200)에 의해 평가될 수 있는 비트스트림 포맷이 설명되었다. 산술적으로-인코딩된 스펙트럼 값의 비트스트림은 상술한 디코딩 알고리즘에 맞도록 인코딩된다.

[0286] 게다가, 일반적으로, 인코딩은 디코딩의 역 연산이어서, 일반적으로 인코더가 상술한 테이블을 이용하는 테이블 룩업(lookup)을 수행하는 것으로 추정될 수 있으며, 이러한 테이블 룩업은 디코더에 의해 수행되는 테이블 룩업과 대략 반대인 것으로 언급되어야 한다. 일반적으로, 디코딩 알고리즘 및/또는 원하는 비트스트림 구문을 알고 있는 당업자는 비트스트림 구문에 정의되고, 산술 디코더에 의해 필요로 되는 데이터를 제공하는 산술 인코더를 쉽게 설계할 수 있을 것이라고 말할 수 있다.

[0287] 10. 구현 대안

[0288] 일부 양태가 장치와 관련하여 설명되었지만, 이들 양태는 또한 상응하는 방법에 대한 설명을 명백히 나타내며, 여기서, 블록 또는 디바이스는 방법 단계 또는 방법 단계의 특징에 상응한다. 유사하게도, 방법 단

계와 관련하여 설명된 양태는 또한 상응하는 장치의 상응하는 블록 또는 항목 또는 특징에 대한 설명을 나타낸다. 방법 단계의 일부 또는 모두는 예컨대, 마이크로프로세서, 프로그램 가능한 컴퓨터 또는 전자 회로와 같은 하드웨어 장치에 의해(또는 이용하여) 실행될 수 있다. 일부 실시예들에서, 가장 중요한 방법 단계 중 일부의 하나 이상은 이와 같은 장치에 의해 실행될 수 있다.

[0289] 발명의 인코딩된 오디오 신호는 디지털 저장 매체 상에 저장될 수 있거나, 무선 전송 매체와 같은 전송 매체 또는 인터넷과 같은 유선 전송 매체 상에서 전송될 수 있다.

[0290] 어떤 구현 요구 사항에 따라, 본 발명의 실시예들은 하드웨어 또는 소프트웨어에서 구현될 수 있다. 이런 구현은 디지털 저장 매체, 예컨대, 플로피 디스크, DVD, 블루레이, CD, ROM, PROM, EPROM, EEPROM 또는 플래시 메모리를 이용하여 실행될 수 있으며, 이들은 전자식 판독 가능한 제어 신호를 저장하여, 각각의 방법이 실행되도록 하는 프로그램 가능한 컴퓨터 시스템과 협력한다 (또는 협력할 수 있다). 그래서, 디지털 저장 매체는 컴퓨터 판독 가능할 수 있다.

[0291] 본 발명에 따른 일부 실시예들은 여기에 설명된 방법 중 하나가 수행되도록 프로그램 가능한 컴퓨터 시스템과 협력할 수 있는 전자식 판독 가능한 제어 신호를 가진 데이터 캐리어를 포함한다.

[0292] 일반적으로, 본 발명의 실시예들은 프로그램 코드를 가진 컴퓨터 프로그램 제품으로서 구현될 수 있으며, 이 프로그램 코드는 컴퓨터 프로그램 제품이 컴퓨터 상에서 실행될 시에 방법 중 하나를 수행하기 위해 동작 가능하다. 프로그램 코드는, 예컨대, 기계 판독 가능한 캐리어 상에 저장될 수 있다.

[0293] 다른 실시예들은, 기계 판독 가능한 캐리어 상에 저장되고, 여기에 설명된 방법 중 하나를 실행하는 컴퓨터 프로그램을 포함한다.

[0294] 그래서, 환언하면, 발명의 방법의 실시예는, 컴퓨터 프로그램이 컴퓨터 상에서 실행될 시에, 여기에 설명된 방법 중 하나를 실행하기 위한 프로그램 코드를 가진 컴퓨터 프로그램이다.

[0295] 그래서, 발명의 방법의 추가 실시예는, 여기에 설명된 방법 중 하나를 실행하기 위한 컴퓨터 프로그램을 기록한 데이터 캐리어 (또는 디지털 저장 매체, 또는 컴퓨터 판독 가능한 매체)이다.

[0296] 그래서, 발명의 방법의 추가 실시예는 여기에 설명된 방법 중 하나를 실행하기 위한 컴퓨터 프로그램을 나타내는 데이터 스트림 또는 신호의 시퀀스이다. 데이터 스트림 또는 신호의 시퀀스는, 예컨대, 데이터 통신 접속을 통해, 예컨대, 인터넷을 통해 전송되도록 구성될 수 있다.

[0297] 추가 실시예는, 여기에 설명된 방법 중 하나를 실행하기 위해 구성되거나 적응되는 처리 수단, 예컨대, 컴퓨터, 또는 프로그램 가능한 논리 디바이스를 포함한다.

[0298] 추가 실시예는 여기에 설명된 방법 중 하나를 실행하기 위한 컴퓨터 프로그램을 설치한 컴퓨터를 포함한다.

[0299] 일부 실시예들에서, 프로그램 가능한 논리 디바이스 (예컨대, 필드 프로그램 가능 게이트 어레이)는 여기에 설명된 방법의 일부 또는 모든 기능을 실행하는데 이용될 수 있다. 일부 실시예들에서, 필드 프로그램 가능 게이트 어레이는 여기에 설명된 방법 중 하나를 실행하기 위해 마이크로프로세서와 협력할 수 있다. 일반적으로, 이들 방법은 바람직하게는 어떤 하드웨어 장치에 의해 실행된다.

[0300] 상술한 실시예들은 단지 본 발명의 원리를 위해 예시한 것이다. 여기에 설명된 배치 및 상세 사항의 수정 및 변형은 당업자에게는 자명한 것으로 이해된다. 그래서, 여기의 실시예의 설명을 통해 제시된 특정 상세 사항에 의해 제한되지 않고, 첨부한 특허청구범위의 범주에 의해서만 제한되는 것으로 의도된다.

[0301] 상술한 것이 특히 상기 특정 실시예에 관련하여 나타내고 설명되었지만, 당업자는 형식 및 상세 사항에서 여러 다른 변경이 이의 정신 및 범위를 벗어나지 않고 행해질 수 있는 것으로 이해할 것이다. 여기에 개시된 광범위한 개념에서 벗어나지 않고 여러 실시예에 적응시키는데 있어 다양한 변경이 행해질 수 있고, 다음의 청구범위에 의해 함축되는 것으로 이해될 수 있다.

[0302] 11. 결론

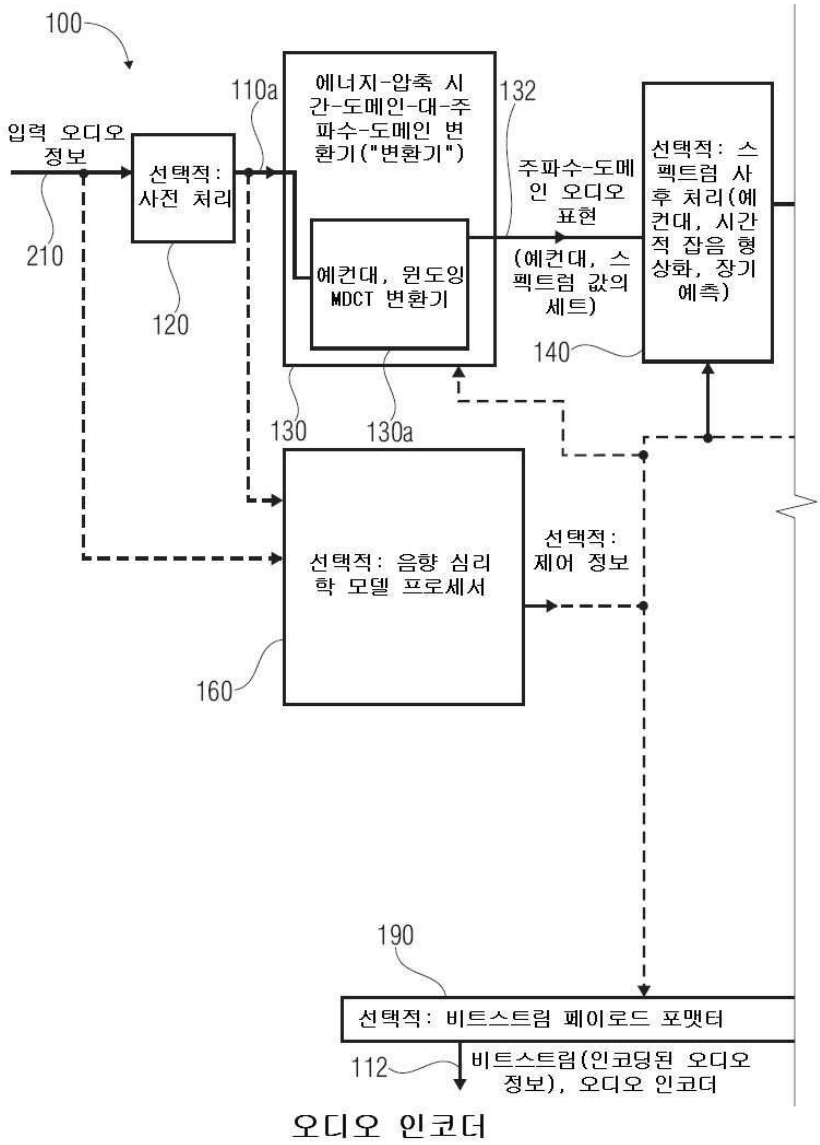
[0303] 결론적으로, 본 발명에 따른 실시예들은 향상된 스펙트럼 잡음없는 코딩 기법을 생성하는 것으로 언급될 수 있다. 새로운 제안에 따른 실시예들은 16894.5 워드에서 900 워드로(ROM) 및 666 워드에서 72 워드로(코어-코더 채널 당 정적 RAM) 메모리 수요의 상당한 감소를 허용한다. 이것은 하나의 실시예에서 대략 43%만큼 완전한 시스템의 데이터 ROM 수요의 감소를 허용한다. 동시에, 코딩 성능은 완전히 유지될 뿐만 아니라, 평균하여 심지어 증가된다. WD3(또는 USAC 초안 표준의 WD3에 따라 제공되는 비트스트림)의 무손실 트랜스코딩이 가능

한 것으로 입증되었다. 따라서, 본 발명에 따른 실시예는 여기에서 설명된 무손실 디코딩을 USAC 초안 표준의 향후 작업 초안에 채택함으로써 획득된다.

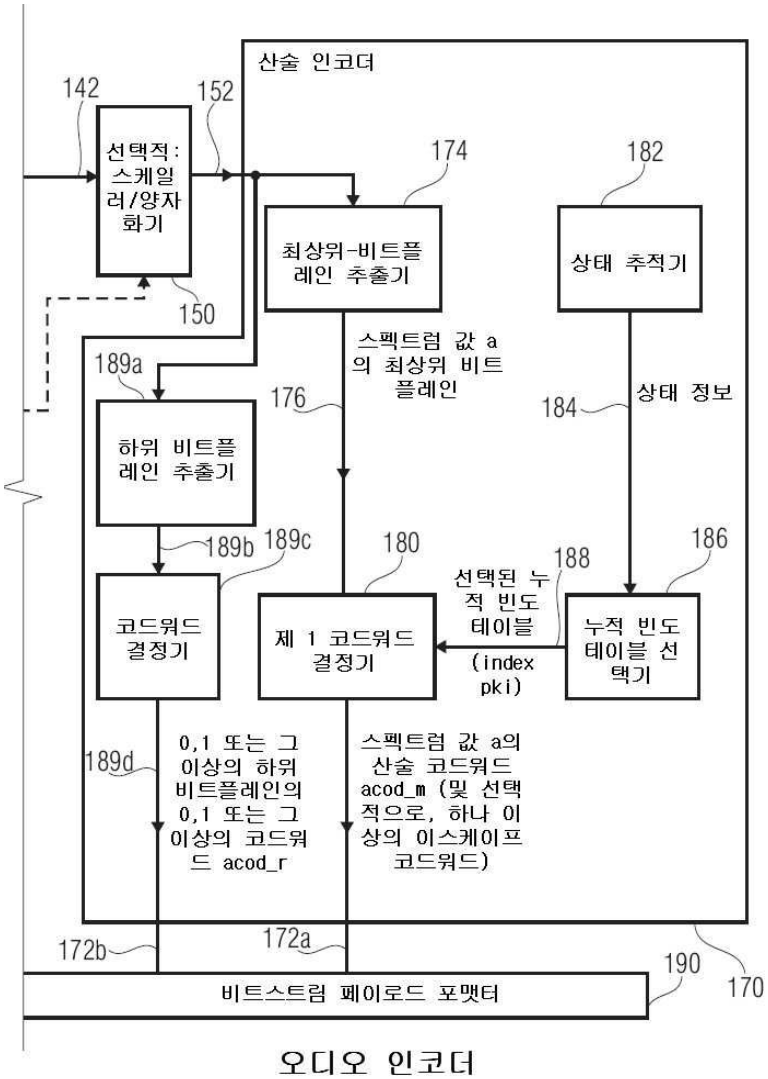
[0304] 요약하면, 실시예에서 제안된 새로운 잡음없는 코딩은, 도 6g에 도시된 바와 같은 비트스트림 요소 "arith_data()"의 구문, 상술하고, 도 5h에 도시된 바와 같은 스펙트럼 잡음없는 코더의 페이로드, 상술한 바와 같은 스펙트럼 잡음없는 코딩, 도 4에 도시된 바와 같은 상태 계산에 대한 컨텍스트, 도 5i에 도시된 바와 같은 정의, 도 5a, 5b, 5c, 5e, 5g, 5h와 관련하여 상술한 바와 같은 디코딩 프로세스, 도 17, 18, 20에 도시된 바와 같은 테이블, 및 도 5d에 도시된 바와 같은 함수 "get_pk"에 대하여, MPEG-USAC 작업 초안에서 수정을 생성시킬 수 있다. 그러나, 대안적으로, 도 20에 따른 테이블 "ari_s_hash"은 도 17의 테이블 "ari_s_hash" 대신에 이용될 수 있고, 도 5f의 함수 "get_pk"는 도 5d에 따른 함수 "get_pk" 대신에 이용될 수 있다.

도면

도면1a

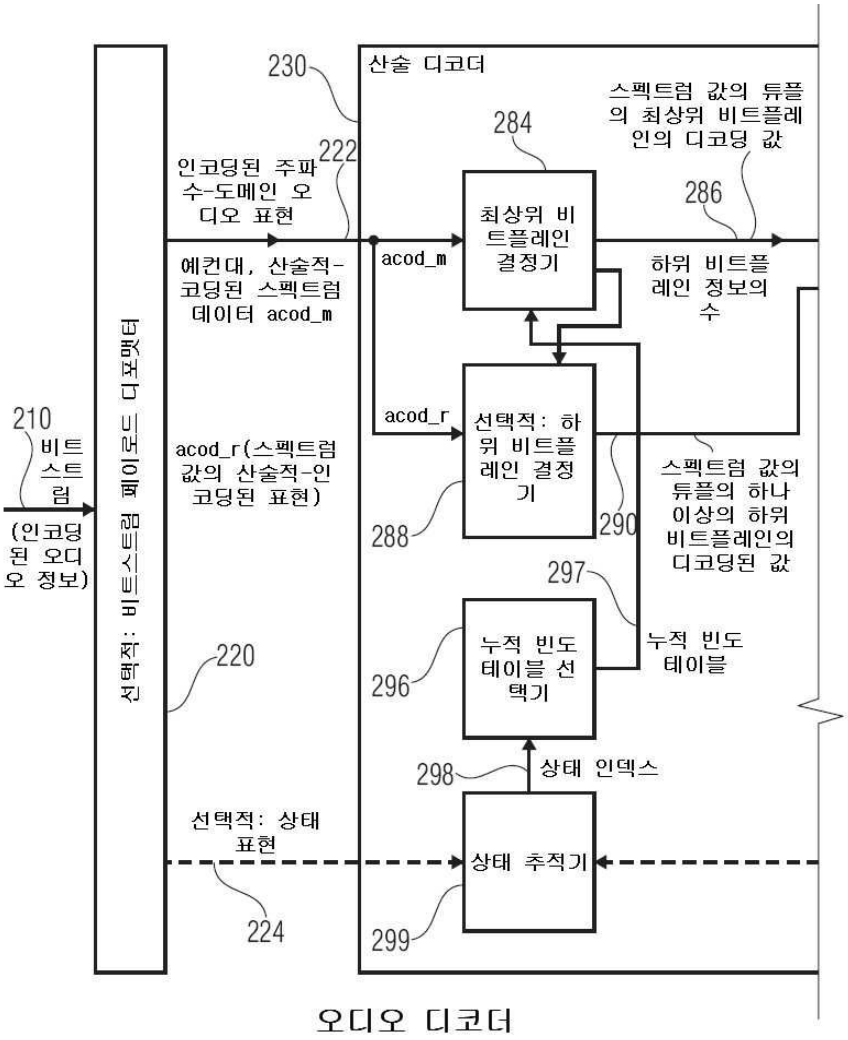


도면1b


산술 인코더
142
152
선택적: 스케일러/양자화기
150
174
최상위-비트플레인 추출기
182
상태 추적기
스펙트럼 값 a의 최상위 비트플레인
176
184
상태 정보
189a
하위 비트플레인 추출기
189b
189c
180
188
186
선택된 누적 빈도 테이블 (index pki)
코드워드 결정기
제 1 코드워드 결정기
누적 빈도 테이블 선택기
189d
0,1 또는 그 이상의 하위 비트플레인의 0,1 또는 그 이상의 코드워드 acod_r
스펙트럼 값 a의 산술 코드워드 acod_m (및 선택적으로, 하나 이상의 이스케이프 코드워드)
172b
172a
190
170
비트스트림 페이로드 포맷터

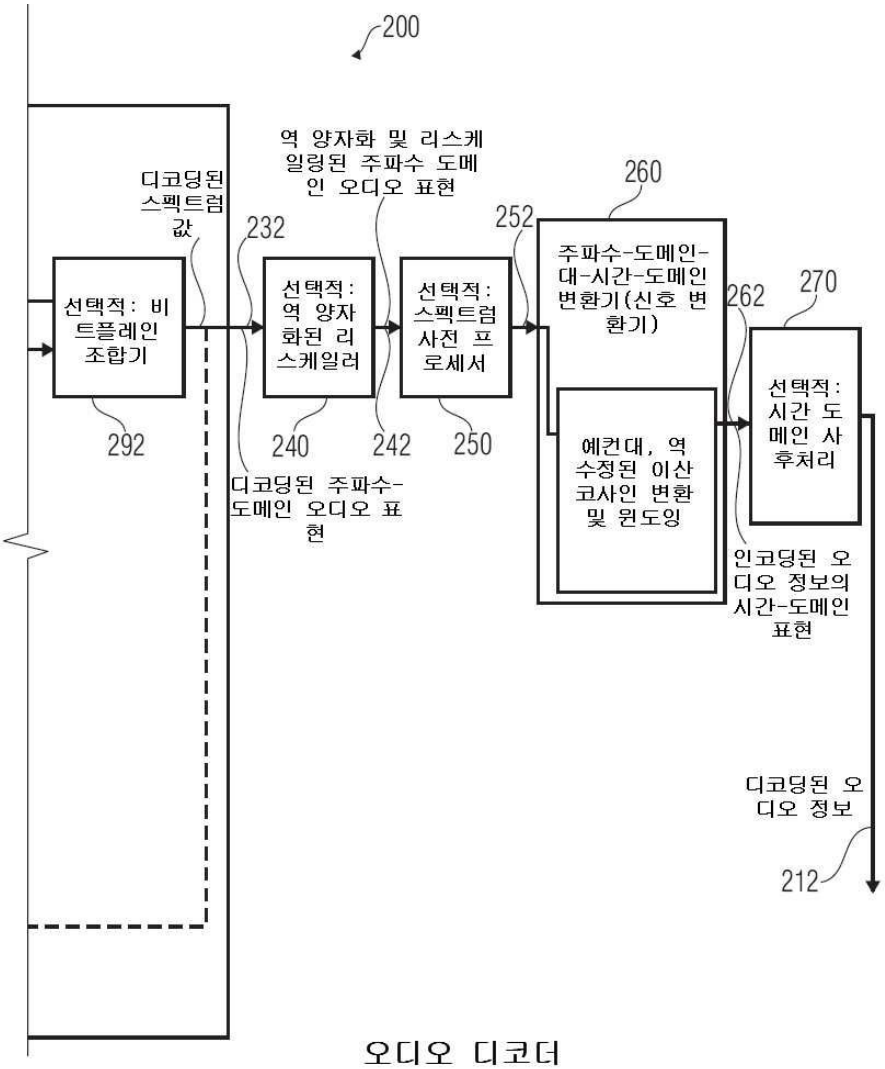
오디오 인코더

도면2a

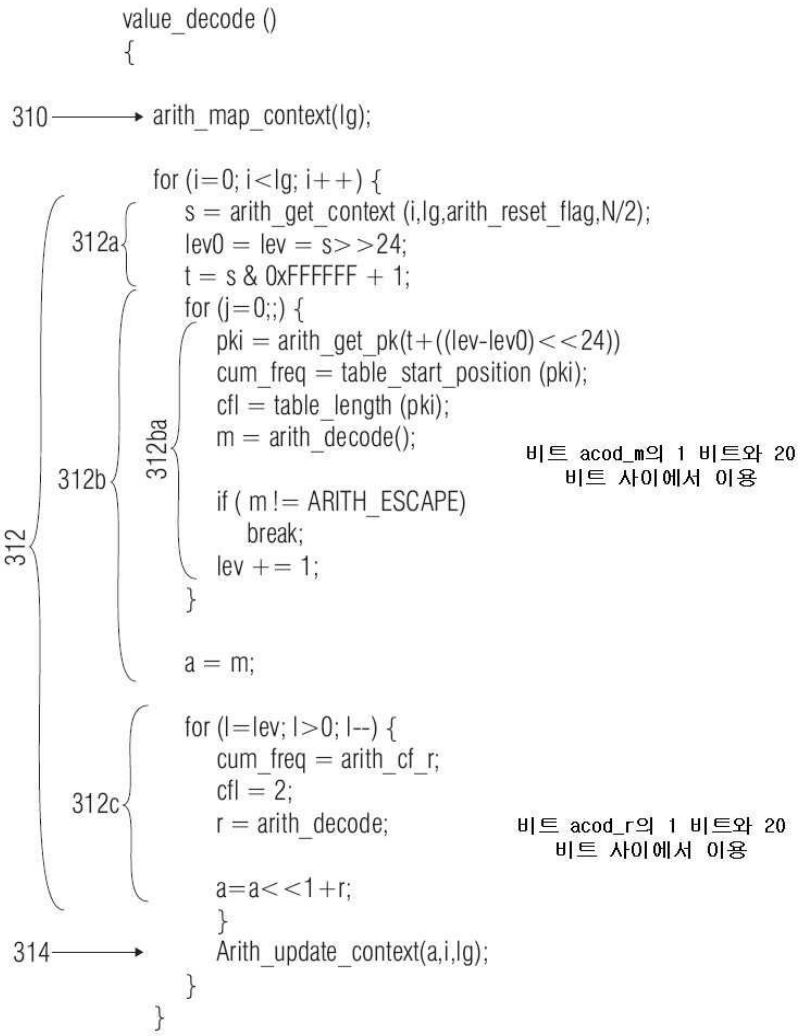

230
산술 디코더
284
스펙트럼 값의 튜플의 최상위 비트플레인의 디코딩 값
222
인코딩된 주파수-도메인 오디오 표현
286
최상위 비트플레인 결정기
acod_m
하위 비트플레인 정보의 수
예컨대, 산술적-코딩된 스펙트럼 데이터 acod_m
acod_r
선택적: 하위 비트플레인 결정기
210
비트스트림
(인코딩된 오디오 정보)
acod_r(스펙트럼 값의 산술적-인코딩된 표현)
288
290
스펙트럼 값의 튜플의 하나 이상의 하위 비트플레인의 디코딩된 값
297
누적 빈도 테이블
296
누적 빈도 테이블 선택기
220
선택적: 비트스트림 페이로드 디포맷터
298
상태 인덱스
선택적: 상태 표현
상태 추적기
224
299

오디오 디코더

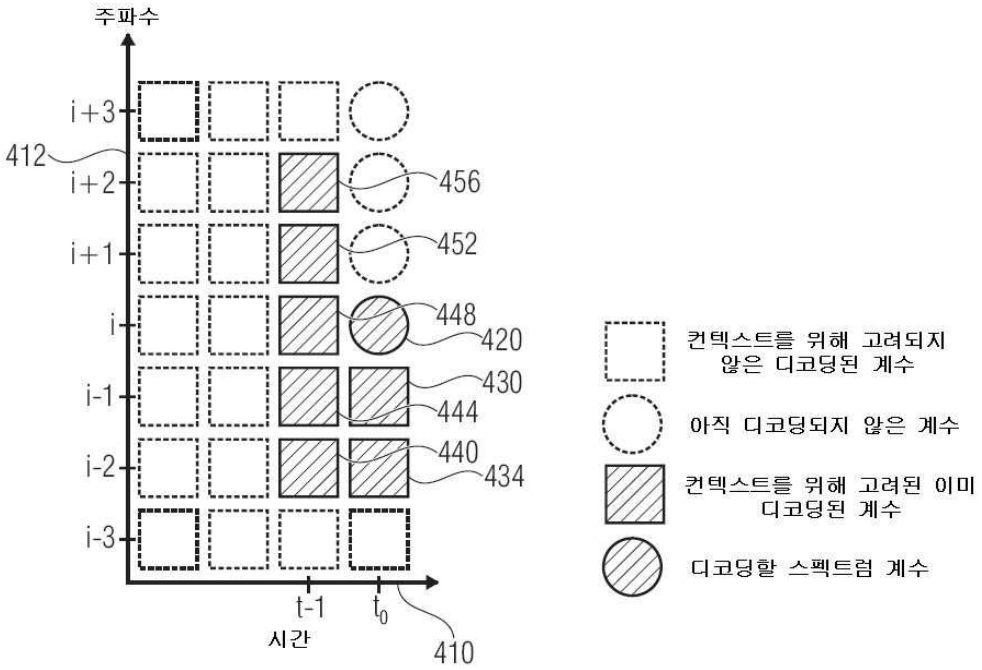
도면2b


200
디코딩된 스펙트럼 값
역 양자화 및 리스케일링된 주파수 도메인 오디오 표현
232
252
260
262
270
선택적: 비트플레인 조합기
선택적: 역 양자화된 리스케일러
선택적: 스펙트럼 사전 프로세서
주파수-도메인-대-시간-도메인 변환기(신호 변환기)
선택적: 시간 도메인 사후처리
292
240
242
250
예컨대, 역 수정된 이산 코사인 변환 및 윈도잉
디코딩된 주파수-도메인 오디오 표현
인코딩된 오디오 정보의 시간-도메인 표현
디코딩된 오디오 정보
212
오디오 디코더

도면3


value_decode ()
{
310
arith_map_context(lg);
for (i=0; i<lg; i++) {
312a
s = arith_get_context (i,lg,arith_reset_flag,N/2);
lev0 = lev = s>>24;
t = s & 0xFFFFFF + 1;
312b
for (j=0;;) {
312ba
pki = arith_get_pk(t+((lev-lev0)<<24))
cum_freq = table_start_position (pki);
cfl = table_length (pki);
m = arith_decode();
비트 acod_m의 1 비트와 20 비트 사이에서 이용
if (m != ARITH_ESCAPE)
break;
lev += 1;
}
a = m;
312c
for (l=lev; l>0; l--) {
cum_freq = arith_cf_r;
cfl = 2;
r = arith_decode;
비트 acod_r의 1 비트와 20 비트 사이에서 이용
a=a<<1+r;
}
314
Arith_update_context(a,i,lg);
}
}
312

도면4

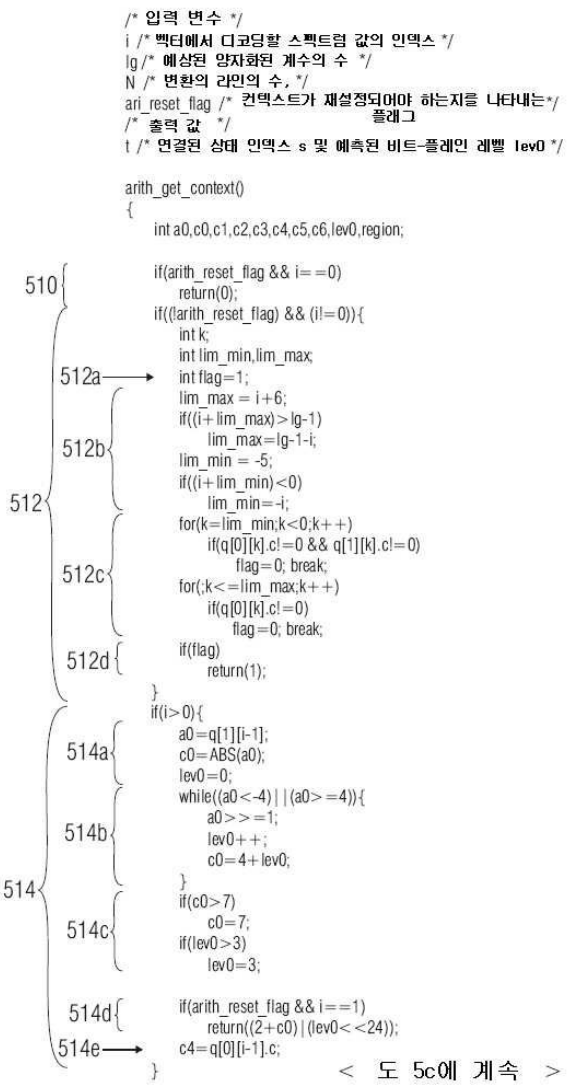


상태 계산을 위한 컨텍스트

도면5a

```
/* 입력 변수 */
lg /* 프레임에서 디코딩할 스펙트럼 계수의 수 */
previous_lg / 이전의 프레임의 스펙트럼 라인의 이전의 수 */

arith_map_context()
{
      v=w=0

      ratio= ((float)previous_lg)/((float)lg);
      for(j=0; j<lg; j++){
            k = (int) ((float)) ((j)*ratio);
            q[0][v++].c = qs[w+k];
      }

      previous_lg=lg;
}
```

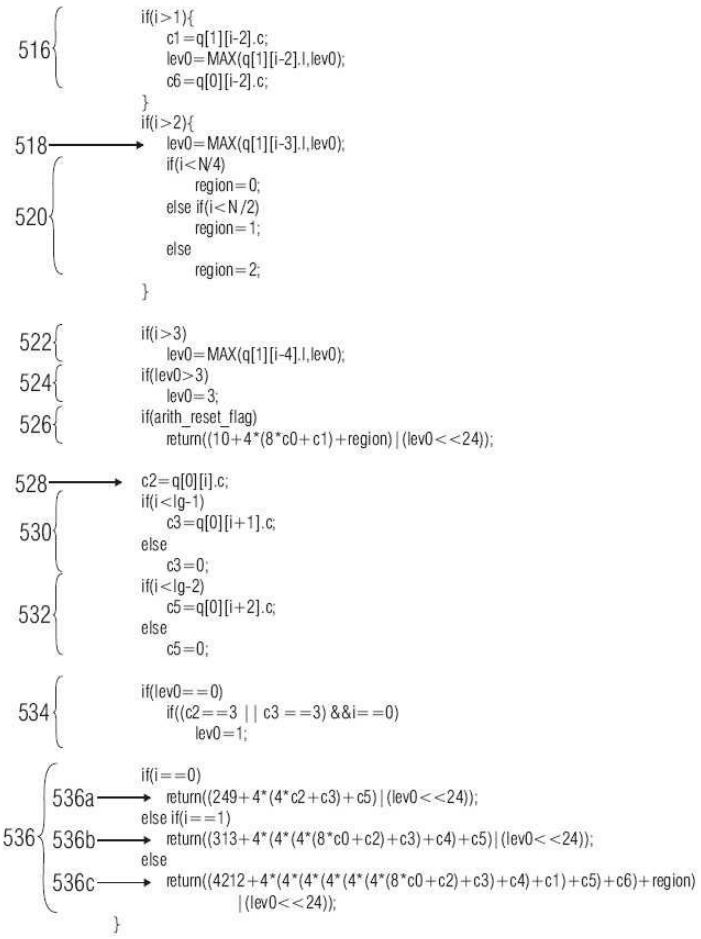
도면5b


/* 입력 변수 */
i /* 벡터에서 디코딩할 스펙트럼 값의 인덱스 */
lg /* 예상된 양자화된 계수의 수 */
N /* 변환의 라인의 수, */
ari_reset_flag /* 컨텍스트가 재설정되어야 하는지를 나타내는*/
/* 출력 값 */ 플래그
t /* 연결된 상태 인덱스 s 및 예측된 비트-플레인 레벨 lev0 */
arith_get_context()
{
int a0,c0,c1,c2,c3,c4,c5,c6,lev0,region;
if(arith_reset_flag && i==0)
return(0);
if((!arith_reset_flag) && (i!=0)){
int k;
int lim_min,lim_max;
int flag=1;
lim_max = i+6;
if((i+lim_max)>lg-1)
lim_max=lg-1-i;
lim_min = -5;
if((i+lim_min)<0)
lim_min=-i;
for(k=lim_min;k<0;k++)
if(q[0][k].c!=0 && q[1][k].c!=0)
flag=0; break;
for(;k<=lim_max;k++)
if(q[0][k].c!=0)
flag=0; break;
if(flag)
return(1);
}
if(i>0){
a0=q[1][i-1];
c0=ABS(a0);
lev0=0;
while((a0<-4)||(a0>=4)){
a0>>=1;
lev0++;
c0=4+lev0;
}
if(c0>7)
c0=7;
if(lev0>3)
lev0=3;
if(arith_reset_flag && i==1)
return((2+c0)|(lev0<<24));
c4=q[0][i-1].c;
}
510
512
512a
512b
512c
512d
514
514a
514b
514c
514d
514e

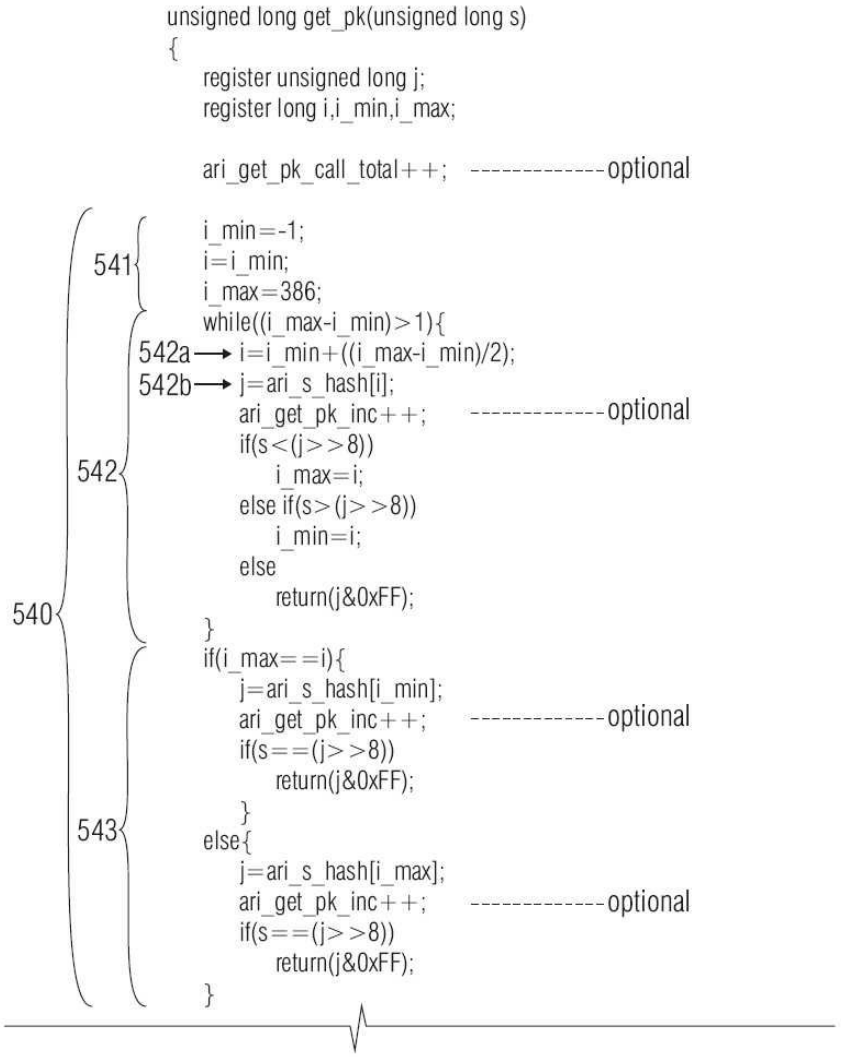
< 도 5c에 계속 >

도면5c

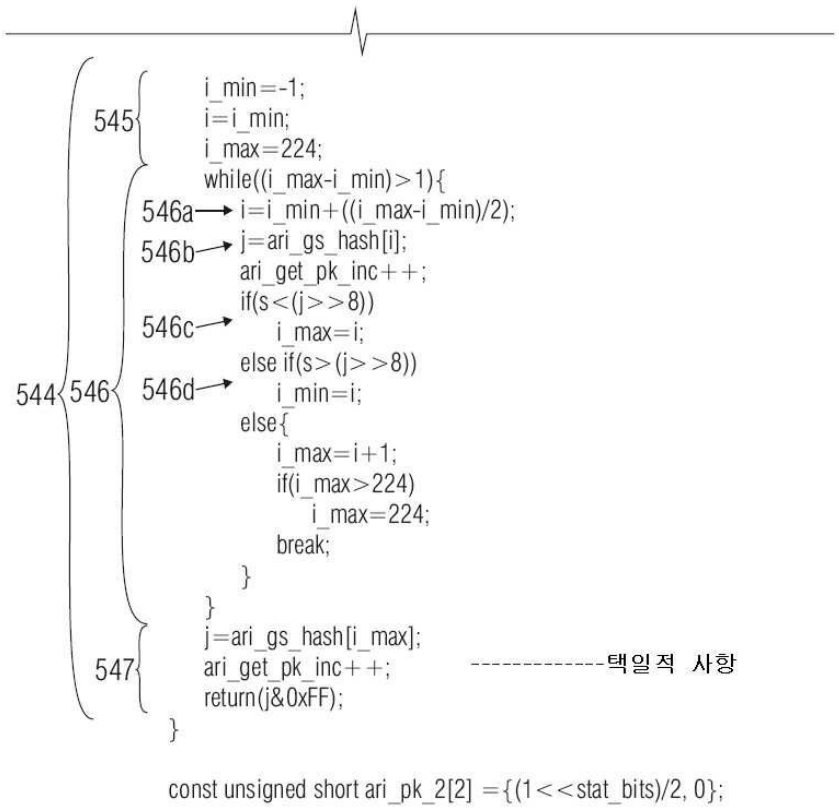
< 도 5b로 부터 >


516
if(i>1){
c1=q[1][i-2].c;
lev0=MAX(q[1][i-2].l,lev0);
c6=q[0][i-2].c;
}
518
if(i>2){
lev0=MAX(q[1][i-3].l,lev0);
520
if(i<N/4)
region=0;
else if(i<N/2)
region=1;
else
region=2;
}
522
if(i>3)
lev0=MAX(q[1][i-4].l,lev0);
524
if(lev0>3)
lev0=3;
526
if(arith_reset_flag)
return((10+4*(8*c0+c1)+region)|(lev0<<24));
528
c2=q[0][i].c;
530
if(i<lg-1)
c3=q[0][i+1].c;
else
c3=0;
532
if(i<lg-2)
c5=q[0][i+2].c;
else
c5=0;
534
if(lev0==0)
if((c2==3 || c3==3) &&i==0)
lev0=1;
536
if(i==0)
536a
return((249+4*(4*c2+c3)+c5)|(lev0<<24));
else if(i==1)
536b
return((313+4*(4*(4*(8*c0+c2)+c3)+c4)+c5)|(lev0<<24));
else
536c
return((4212+4*(4*(4*(4*(4*(8*c0+c2)+c3)+c4)+c1)+c5)+c6)+region)
|(lev0<<24));
}

도면5da


unsigned long get_pk(unsigned long s)
{
register unsigned long j;
register long i,i_min,i_max;
ari_get_pk_call_total++; ----------optional
i_min=-1;
i=i_min;
i_max=386;
while((i_max-i_min)>1){
i=i_min+((i_max-i_min)/2);
j=ari_s_hash[i];
ari_get_pk_inc++; ----------optional
if(s<(j>>8))
i_max=i;
else if(s>(j>>8))
i_min=i;
else
return(j&0xFF);
}
if(i_max==i){
j=ari_s_hash[i_min];
ari_get_pk_inc++; ----------optional
if(s==(j>>8))
return(j&0xFF);
}
else{
j=ari_s_hash[i_max];
ari_get_pk_inc++; ----------optional
if(s==(j>>8))
return(j&0xFF);
}
541
542a
542b
542
540
543

도면5db


i_min=-1;
545
i=i_min;
i_max=224;
while((i_max-i_min)>1){
546a
i=i_min+((i_max-i_min)/2);
546b
j=ari_gs_hash[i];
ari_get_pk_inc++;
if(s<(j>>8))
546c
i_max=i;
else if(s>(j>>8))
544
546
546d
i_min=i;
else{
i_max=i+1;
if(i_max>224)
i_max=224;
break;
}
}
j=ari_gs_hash[i_max];
547
ari_get_pk_inc++;
----------택일적 사항
return(j&0xFF);
}
const unsigned short ari_pk_2[2] ={(1<<stat_bits)/2, 0};

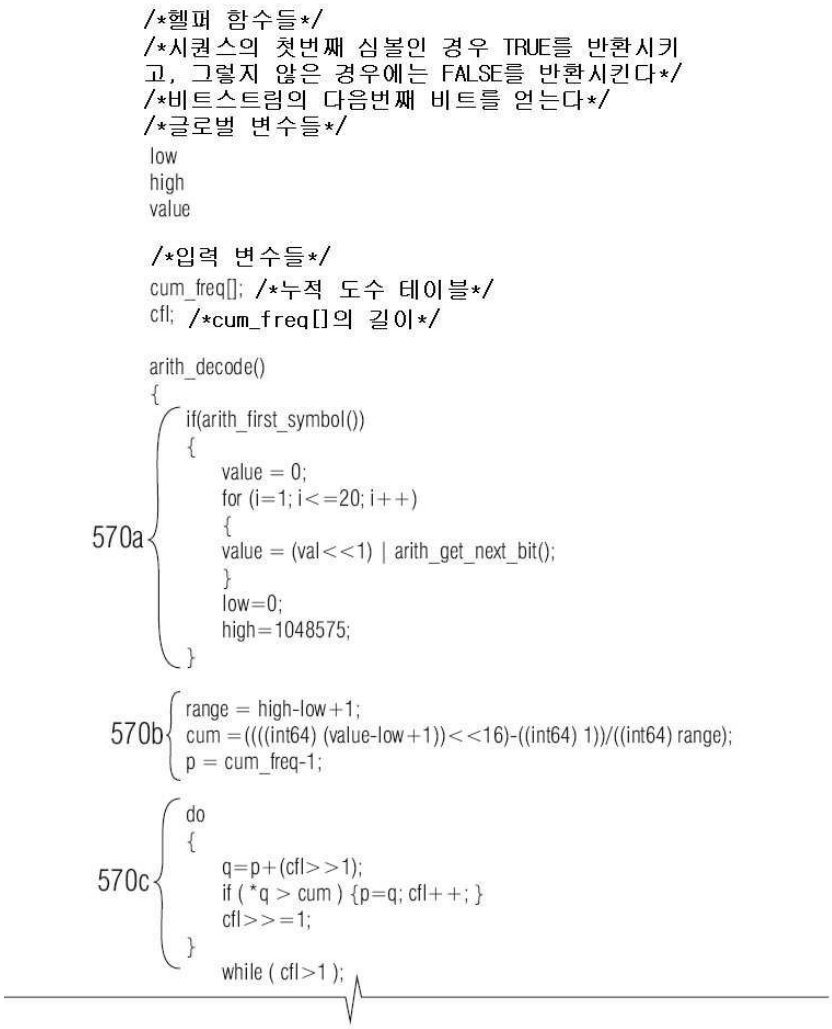
도면5e

```
/*  입력 변수   */
s  /*  컨텍스트의 상태   */
/*  출력 값    */
pki /*      확률 모델의 인덱스          */

arith_get_pk(s)
{
    register unsigned long i,j;

    for (i=0;i<387;i++)
    {
        j=ari_s_hash[i];
        if ( (j>>8)==s ) return j&255;
    }
    for (i=0;i<225;i++)
    {
        j=ari_gs_hash[i];
        if ( s<(j>>8) ) return j&255;
    }
    return j&255;
}
```

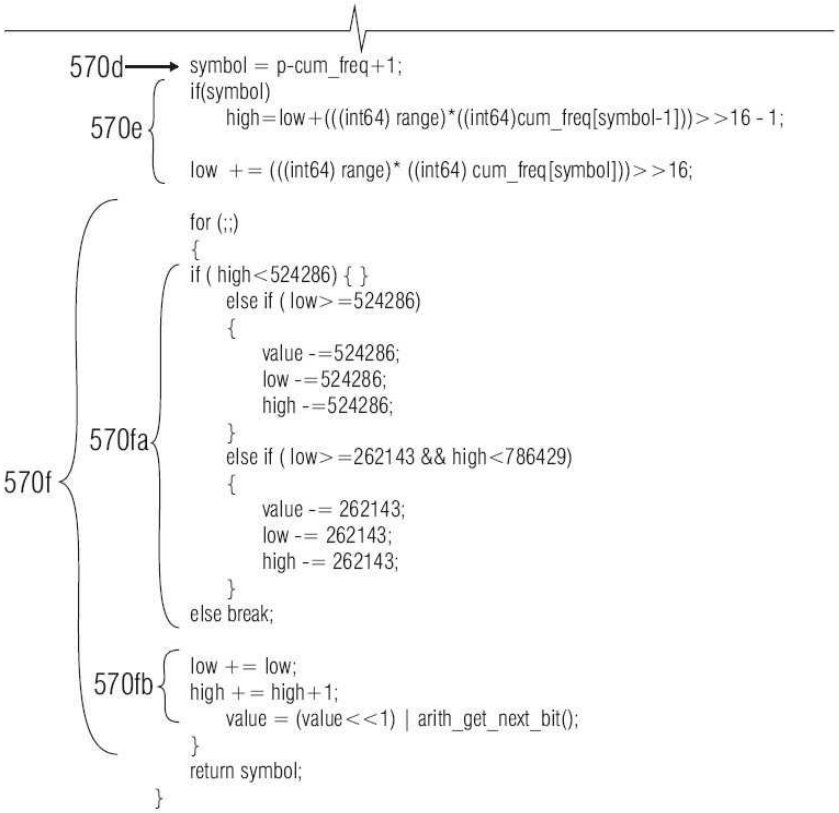
도면5f

```
unsigned long get_pk(unsigned long s)
{
    register unsigned longlong j;
    register unsigned long  i;

    for (i=0;i<387;i++)
    {
        j=ari_s_hash[i];
        if ( (j>>8)==s )
            return j&0xFF;
    }
    for(i=0;i<225;i++){
        j=ari_gs_hash[i];
        if ( s<(j>>8) ) return j&0xFF;
    }
    return(j&0xFF);
}
```

도면5ga


/*헬퍼 함수들*/
/*시퀀스의 첫번째 심볼인 경우 TRUE를 반환시키고, 그렇지 않은 경우에는 FALSE를 반환시킨다*/
/*비트스트림의 다음번째 비트를 얻는다*/
/*글로벌 변수들*/
low
high
value
/*입력 변수들*/
cum_freq[]; /*누적 도수 테이블*/
cfl; /*cum_freq[]의 길이*/
arith_decode()
{
if(arith_first_symbol())
{
value = 0;
for (i=1; i<=20; i++)
{
value = (val<<1) | arith_get_next_bit();
}
low=0;
high=1048575;
}
570a
range = high-low+1;
cum =((((int64) (value-low+1))<<16)-((int64) 1))/((int64) range);
p = cum_freq-1;
570b
do
{
q=p+(cfl>>1);
if (*q > cum) {p=q; cfl++; }
cfl>>=1;
}
while (cfl>1);
570c

도면5gb


570d
symbol = p-cum_freq+1;
if(symbol)
570e
high=low+(((int64) range)*((int64)cum_freq[symbol-1]))>>16 - 1;
low += (((int64) range)* ((int64) cum_freq[symbol]))>>16;
570f
for (;;)
{
if (high<524286) { }
570fa
else if (low>=524286)
{
value -=524286;
low -=524286;
high -=524286;
}
else if (low>=262143 && high<786429)
{
value -= 262143;
low -= 262143;
high -= 262143;
}
else break;
570fb
low += low;
high += high+1;
value = (value<<1) | arith_get_next_bit();
}
return symbol;
}

도면5h

```
/*입력 변수들*/
a /*디코딩되고 양자화된 스펙트럼 계수*/
i /*디코딩될 양자화된 스펙트럼 계수의 인덱스*/
lg /*프레임에서의 계수들의 갯수*/

arith_update_context()
{   int a0;

580 → q[1][i]=a0=a;
      q[1][i].l=0;
      while(ABS(a0)>4){                    582
          a0=a0>>1;
          q[1][i].l++;
      }
      if(a==0)                             584
          q[1][i].c=0;
      else if(ABS(a)<=1)
          q[1][i].c=1;
      else if(ABS(a)<=3)
          q[1][i].c=2;
      else
          q[1][i].c=3;

      if(i==lg && core_mode==1){           586
          ratio= ((float) lg)/((float)1024);
          for(j=0; j<1024; j++){
              k = (int) ((float) j*ratio);
              qs[j] = q[1][k].c ;
          }
          previous_lg = 1024;
      }
      if(i==lg/4 && core_mode==0){         588
          for(j=0; j<MIN(lg,1024; j++){
              qs[j] = q[1][j].c;
          }
          previous_lg = MIN(1024,lg);
      }
}
```

도면5i

정의

a	디코딩될 양자화된 계수
m	디코딩될 양자화된 스펙트럼 계수의 최상위 2비트 와이즈 플레인
r	디코딩될 양자화된 스펙트럼 계수의 최상위 2비트 와이즈 플레인
lev	나머지 비트플레인들의 레벨. 이것은 최상위 2 비트 와이즈 플레인보다 하위에 있는 비트플레인들의 개수에 대응한다
lev0	예상된 비트플레인 레벨
arith_s_hash[]	누적 도수 테이블 인덱스 pki에 콘텍스트의 상태들을 맵핑하는 해쉬 테이블
arith_gs_hash[]	누적 도수 테이블 인덱스 pki에 콘텍스트의 상태들의 그룹을맵핑하는 해쉬 테이블
arith_cf_m[pki][9]	최상위 2비트 와이즈 플레인 m과 ARITH_ESCAPE 심볼에 대한 누적 도수들의 모델들
arith_cf_r []	최하위 비트 플레인 심볼 r에 대한 누적 도수들
previous_lg	산술 디코더에 의해 이전에 디코딩된 전달된 스펙트럼 계수들의 갯수
N	윈도우 길이. AAC의 경우, 이것은 window_sequence로부터 유도되며(섹션 6.8.3.1 참조), TCX의 경우 N=2.lg0이다.
q[2][]	디코딩될 스펙트럼 계수에 대한 현재 콘텍스트
qs[]	다음 프레임을 위해 저장된 과거 콘텍스트
arith_reset_flag	스펙트럼 무잡음 콘텍스트가 재설정되어야 하는지를 표시하는 플래그

도면6a

```
usac_raw_data_block ()
{
    single_channel_element (); and/or
    channel_pair_element ();
}
```

도면6b

single_channel_element()의 구문

구문	비트들의 갯수	연상기호
single_channel_element() { core_mode	1	uimsbf

if (core_mode == 1) {
lpd_channel_stream();
}
else {
fd_channel_stream();
}		

}		

도면6c

channel_pair_element()의 구문

구문	비트들의 갯수	연상기호
channel_pair_element() {		
core_mode0	1	uimsbf
core_mode1	1	uimsbf
ics_info();	택일적 사항: 두 개의 채널들에 대한 공통적인 ics_info	

if (core_mode0 == 1) {
lpd_channel_stream();
}
else {
fd_channel_stream();
}		

if (core_mode1 == 1) {
lpd_channel_stream();
}
else {
fd_channel_stream();
}		

}		

도면6d

ics_info()의 구문

구문	비트들의 갯수	연상기호	
ics_info()			
{			
window_length;	1	uimsbf	
if(window_length !=0) {			
transform_length;	1	uimsbf	
}			
else {			
transform_length=0;			
}			
window_shape;	1	uimsbf	
if (window_length !=0 && transform_length !=0){			
max_sfb;	4	uimsbf	
scale_factor_grouping;	7	uimsbf	
}			택일적 사항
else {			
max_sfb;	6	uimsbf	
}			
}			

도면6e

fd_channel_stream()의 구문

구문	비트들의 갯수	연상기호
fd_channel_stream()		
{		
global_gain;	8	uimsbf
ics_info(); (채널 쌍 엘리먼트내에 포함 되지 않는 경우)		
scale_factor_data ();		
ac_spectral_data ();		
}		

도면6f

ac_spectral_data()의 구문

구문	비트들의 갯수	연상기호
ac_spectral_data() { **arith_reset_flag**	1	uimsbf

for (win=0; win<num_windows; win++){
arith_data(num_bands, arith_reset_flag)
}
}		

도면6g

arith_data()의 구문

구문	비트들의 갯수	연상기호
Arith_data(lg, arith_reset_flag) {		
660 arith_map_context(lg);		

for (i=0; i<lg; i++) {
s = arith_get_context (i,lg,arith_reset_flag,N/2);
lev0 = lev = s>>24;
t = s & 0xFFFFFF + 1;
for (j=0;;) {		

662 pki = arith_get_pk(t+((lev-lev0)<<24))		
663 **acod_m**[pki][m]	1..20	VIclbf
664 if (m != ARITH_ESCAPE) break; lev +=1;		

}
a=m;
for (l=lev; l>0; l--) {		

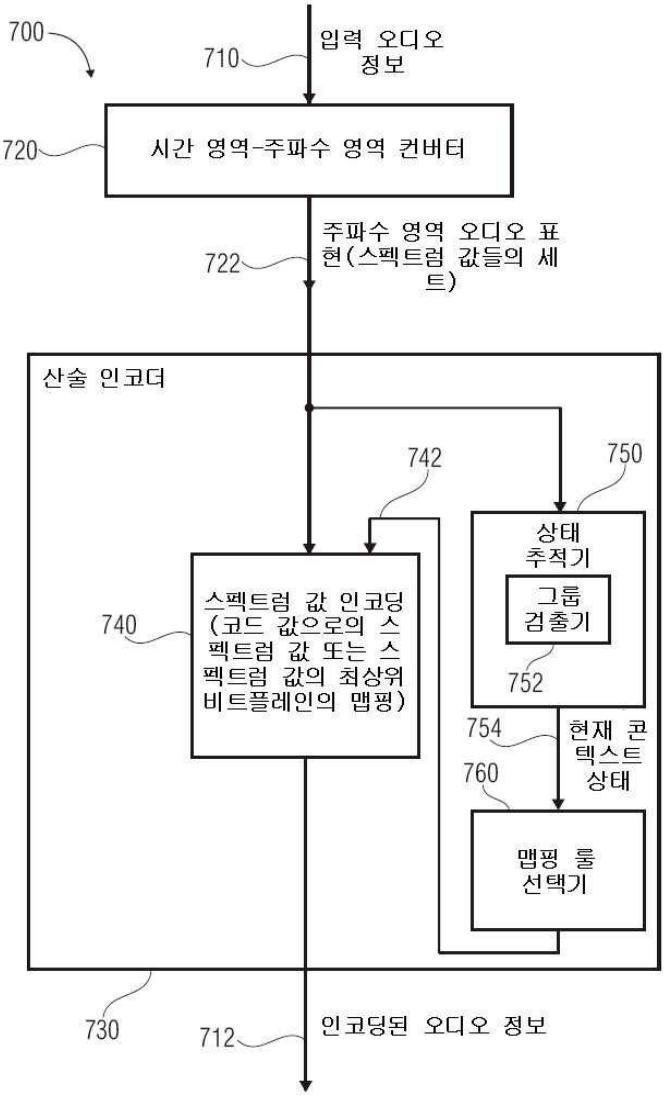
acod_r[r]	1..20	vlclbf

a=a<<1+r;
}
668 Arith_update_context(a,i,lg);
}
}		

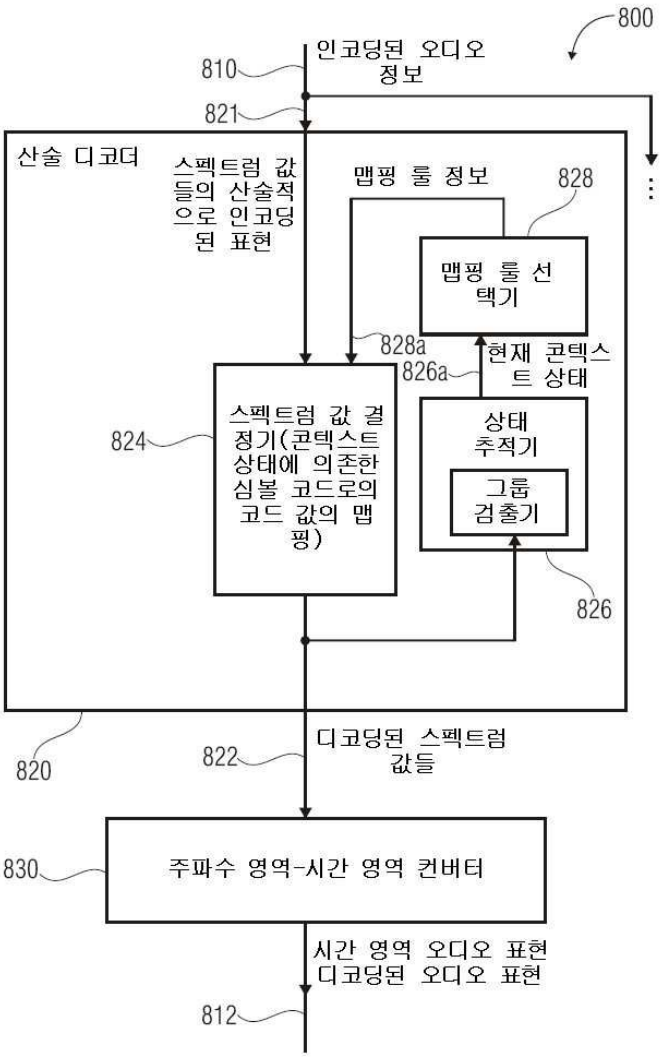
도면6h

정의	
arith_data()	스펙트럼 무잡음 코더 데이터를 디코딩하기 위한 데이터 엘리먼트
arith_reset_flag	스펙트럼 무잡음 콘텍스트가 재설정되어야 하는지를 표시하는 플래그
acod _cf_m[pki][a]	양자화된 스펙트럼 계수의 최상위 2비트 와이즈 플레인 a의 산술 디코딩을 위해 필요한 산술 코드워드
arith_cf_r[]	양자화된 스펙트럼 계수의 잔여 비트플레인들의 산술 디코딩을 위해 필요한 산술 코드워드
도움 구성요소들	
a	디코딩될 스펙트럼 양자화된 계수
m	디코딩될 양자화된 스펙트럼 계수의 최상위 2비트 와이즈 플레인
r	디코딩될 양자화된 스펙트럼 계수의 최상위 2비트 와이즈 플레인
N	윈도우 길이. AAC의 경우, 이것은 window_sequence로부터 유도되며(섹션 6.8.3.1 참조), TCX의 경우 N=2.lg이다.
lg	디코딩될 양자화된 계수들의 갯수
i	프레임 내에서 디코딩될 양자화된 계수들의 인덱스
pki	디코딩 a를 위한 산술 디코더에 의해 이용된 누적 도수 테이블의 인덱스
arith_get_pk ()	코드워드 acod_ng[pki][a]를 디코딩하는데 필요한 누적 도수 테이블의 인덱스 pki를 반환하는 함수
t	콘텍스트의 상태
arith_get_context ()	콘텍스트의 상태를 반환하는 함수
lev0	예상된 비트플레인 레벨
s	예상된 비트플레인 레벨 lev0과 결합된 콘텍스트의 상태
lev	최상위 2비트 와이즈 플레인을 초과하여 디코딩할 비트플레인들의 레벨
ARITH_ESCAPE	예측된 비트플레인 레벨 lev0을 초과하여 디코딩할 추가적인 비트플레인들을 표시하는 탈출 심볼

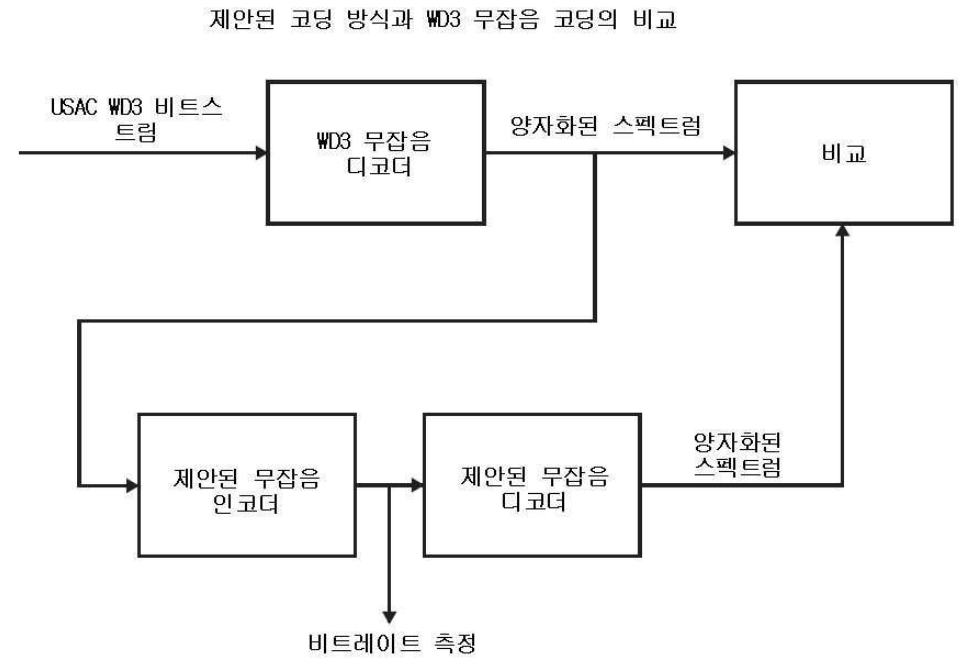
도면7


700
710
입력 오디오 정보
720
시간 영역-주파수 영역 컨버터
722
주파수 영역 오디오 표현(스펙트럼 값들의 세트)
산술 인코더
742
750
상태 추적기
그룹 검출기
752
740
스펙트럼 값 인코딩 (코드 값으로의 스펙트럼 값 또는 스펙트럼 값의 최상위 비트플레인의 맵핑)
754
현재 콘텍스트 상태
760
맵핑 룰 선택기
730
712
인코딩된 오디오 정보

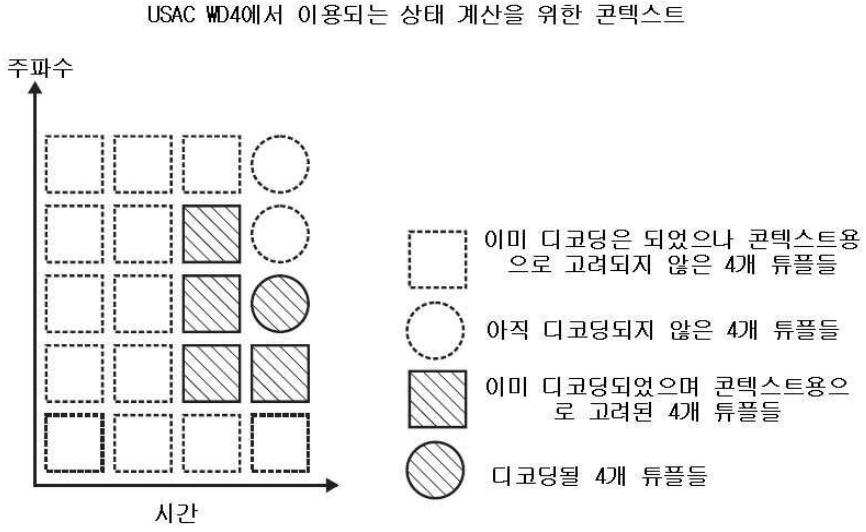
도면8


800
인코딩된 오디오 정보
810
821
산술 디코더
스펙트럼 값들의 산술적으로 인코딩된 표현
맵핑 룰 정보
828
맵핑 룰 선택기
828a
826a
현재 콘텍스트 상태
스펙트럼 값 결정기(콘텍스트 상태에 의존한 심볼 코드로의 코드 값의 맵핑)
824
상태 추적기
그룹 검출기
826
820
822
디코딩된 스펙트럼 값들
830
주파수 영역-시간 영역 컨버터
시간 영역 오디오 표현
디코딩된 오디오 표현
812

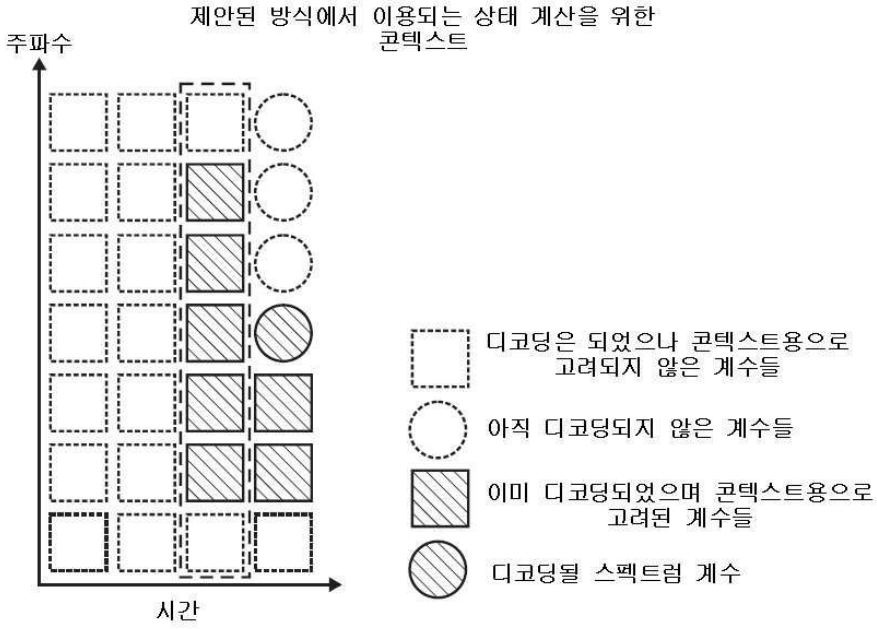
도면9


제안된 코딩 방식과 WD3 무잡음 코딩의 비교
USAC WD3 비트스트림
WD3 무잡음 디코더
양자화된 스펙트럼
비교
제안된 무잡음 인코더
제안된 무잡음 디코더
양자화된 스펙트럼
비트레이트 측정

도면10a


USAC WD4에서 이용되는 상태 계산을 위한 콘텍스트
주파수
시간
이미 디코딩은 되었으나 콘텍스트용으로 고려되지 않은 4개 튜플들
아직 디코딩되지 않은 4개 튜플들
이미 디코딩되었으며 콘텍스트용으로 고려된 4개 튜플들
디코딩될 4개 튜플들

도면10b



도면11a

테이블 명칭	설명	데이터 단위	메모리(32비트의 워드들)
arith_cf_ng_hash[128]	콘텍스트를 확률 모델 인덱스에 맵핑하는 해쉬 테이블	워드	128
arith_cf_ng[32][545]	확률 분포 모델 각각에 대한 그룹들의 누적 도수들	1/2 워드	8720
egroups[8][8][8][8]	4개 튜플의 그룹 인덱스	1/2 워드	2048
dgvectors[4*4096]	4개 튜플에 대한 그룹 인덱스 및 엘리먼트 인덱스 맵핑	1/4 워드	4096
dgroups[544]	dgvectors에서의 그룹 및 오프셋의 카디널에 대한 그룹 인덱스 맵핑	워드	544
arith_cf_ne[2701]	엘리먼트 인덱스 심볼의 누적 도수들	1/2 워드	1350.5
arith_cf_r[16]	최하위 비트 플레인들의 누적 도수들	1/2 워드	8
총계			16894.5

USAC WD4 산술 코딩 방식에서 이용된 테이블들

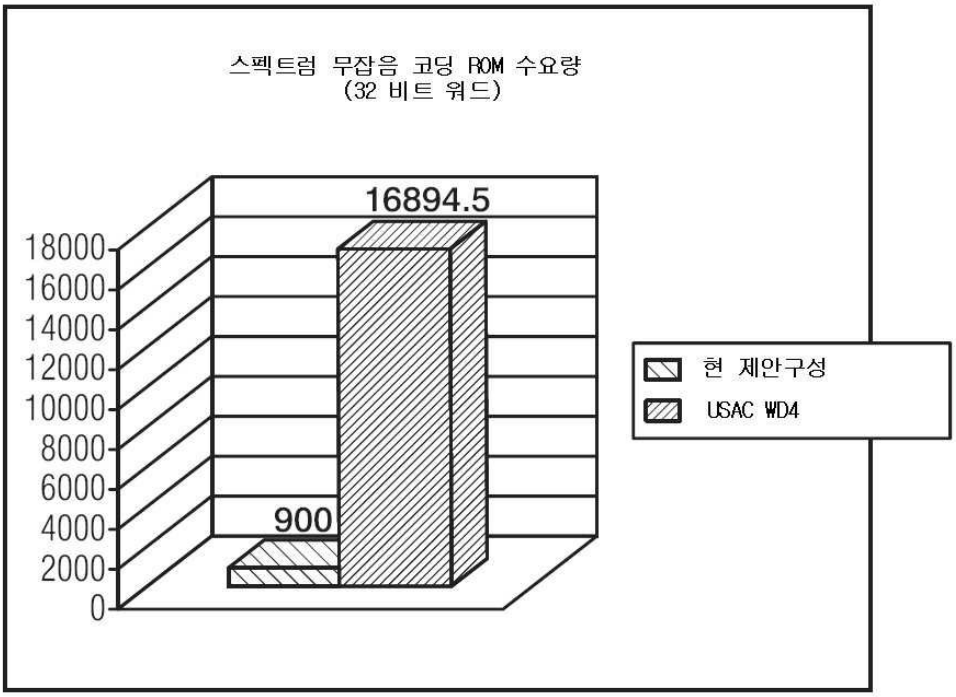
도면11b

제안된 코딩 방식에서 이용된 테이블들

테이블 명칭	설명	데이터 단위	메모리(32비트의 워드들)
arith_s_hash[387]	누적 도수 테이블에 콘텍스트의 상태들을 맵핑하는 해쉬 테이블	워드	387
arith_gs_hash[225]	누적 도수 테이블에 콘텍스트의 상태들의 그룹을 맵핑하는 해쉬 테이블	워드	225
arith_cf_m[64][9]	최상위 2비트 와이즈 플레인 m 및 ARITH_ESCAPE 심볼에 대한 누적 도수들의 모델들	1/2 워드	288
총계			900

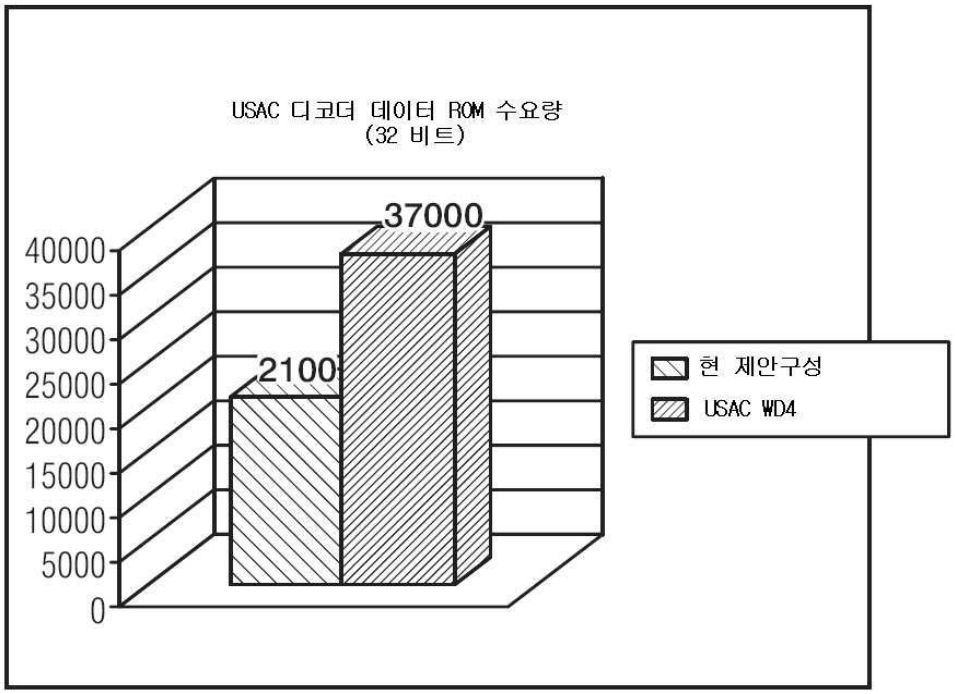
도면12a

WD4에서의 무잡음 코딩 방식 및 제안된 무잡음 코딩 방식의 ROM 수요량



도면12b

현 제안구성 및 WD4에 따른 총체적인 USAC 디코더 데이터 ROM 수요량



도면13a

새로운 제안구성과 WD 산술 코더를 이용한 USAC 코더에 의해 생성된 평균 비트레이트

동작모드	WD (kbit/s)	새로운 제안 (kbit/s)	트랜스코딩 후 차이(kbit/s)	트랜스코딩 후 차이 (총 비트레이트의 %)
테스트 1, 64kbps 스테레오	64.00	63.34	-0.66	-1.04
테스트 2, 32kbps 스테레오	32.00	31.66	-0.34	-1.05
테스트 3, 24kbps 스테레오	24.00	23.73	-0.27	-1.11
테스트 4, 20kbps 스테레오	20.00	19.78	-0.22	-1.11
테스트 5, 16kbps 스테레오	16.00	15.82	-0.18	-1.10
테스트 6, 24kbps 모노	24.00	23.68	-0.32	-1.32
테스트 7, 20kbps 모노	20.00	19.72	-0.28	-1.39
테스트 8, 16kbps 모노	16.00	15.79	-0.21	-1.31
테스트 9, 12kbps 모노	12.00	11.86	-0.14	-1.19

도면13b

USAC WD3 및 새로운 제안구성의 비트저장소 제어

동작모드

비트저장소 제어

동작모드	비트저장소 제어					
	새로운 제안구성			WD		
	min	max	avg	min	max	avg
테스트 1, 64kbps 스테레오	3653	9557	8137	2314	9557	7018
테스트 2, 32kbps 스테레오	1808	4505	4196	581	4505	3530
테스트 3, 24kbps 스테레오	1538	4704	4408	957	4704	3871
테스트 4, 20kbps 스테레오	2367	4864	4600	712	4864	3854
테스트 5, 16kbps 스테레오	2712	5006	4804	724	5006	4234
테스트 6, 24kbps 모노	2185	4704	4457	1002	4704	3927
테스트 7, 20kbps 모노	2599	4864	4630	1192	4864	3935
테스트 8, 16kbps 모노	2820	5006	4876	1434	5006	4450
테스트 9, 12kbps 모노	3529	5184	5081	2256	5184	4787

도면14

USAC WD3 및 새로운 제안구성의 평균 비트레이트

동작모드	평균 비트레이트(kbit/s)					
	새로운 제안구성			WD		
	FD 모드	wLPT 모드	총계	FD 모드	wLPT 모드	총계
테스트 1, 64kbps 스테레오	53.73	---	53.73	54.40	---	54.40
테스트 2, 32kbps 스테레오	25.31	26.34	25.60	25.80	26.61	26.02
테스트 3, 24kbps 스테레오	18.27	19.17	18.50	18.66	19.40	18.85
테스트 4, 20kbps 스테레오	15.50	15.93	15.61	15.83	16.12	15.90
테스트 5, 16kbps 스테레오	12.45	12.60	12.52	12.80	12.73	12.77
테스트 6, 24kbps 모노	19.94	19.51	19.73	20.41	19.42	20.15
테스트 7, 20kbps 모노	16.15	15.91	16.08	16.56	16.12	16.45
테스트 8, 16kbps 모노	13.02	12.59	12.81	13.45	12.73	13.09
테스트 9, 12kbps 모노	9.35	9.66	9.51	9.68	9.71	9.70

도면15

USAC의 프레임 단위의 최소, 최대 및 평균 비트레이트

동작모드	최소 비트레이트 (kbit/s)	최대 비트레이트 (kbit/s)	평균 비트레이트 (kbit/s)
테스트 1, 64kbps 스테레오	15.26	101.79	63.34
테스트 2, 32kbps 스테레오	13.13	48.61	31.66
테스트 3, 24kbps 스테레오	11.69	36.58	23.73
테스트 4, 20kbps 스테레오	3.09	30.94	19.78
테스트 5, 16kbps 스테레오	4.02	26.47	15.82
테스트 6, 24kbps 모노	1.47	37.35	23.68
테스트 7, 20kbps 모노	1.38	31.13	19.72
테스트 8, 16kbps 모노	11.40	24.64	15.79
테스트 9, 12kbps 모노	8.72	18.91	11.86

도면16

프레임 단위의 최상의 경우 및 최악의 경우

동작모드	최상의 경우		최악의 경우	
	(bit/s)	(%)	(bit/s)	(%)
테스트 1, 64kbps 스테레오	-30.87	-33.06	6.14	9.07
테스트 2, 32kbps 스테레오	-10.33	-28.63	2.17	6.77
테스트 3, 24kbps 스테레오	-11.86	-30.75	1.85	7.71
테스트 4, 20kbps 스테레오	-7.45	-30.27	1.67	8.36
테스트 5, 16kbps 스테레오	-5.43	-27.89	1.50	9.42
테스트 6, 24kbps 모노	-17.06	-45.83	1.25	4.36
테스트 7, 20kbps 모노	-15.86	-41.46	0.88	3.38
테스트 8, 16kbps 모노	-4.75	-24.85	1.11	7.31
테스트 9, 12kbps 모노	-3.95	-26.33	0.82	6.99

도면17a

```
 /*
  엔트로피:
  fu mem.: 1.2792 비트 (100.00%)
  no mem.: 1.6289 비트 (127.34%)
  스플리트: 1.2971 비트 (101.40%)
 */
/* 1224개 상태들, 엔트로피 증가: 0.000384*/
/*최종 엔트로피 : 1.297556*/
/*총 상태들 = 612;*/
/*중요 상태들 = 387;*/
/*의사 상태들 = 225;*/
/*확률 모델들 = 64;*/

unsigned long long ari_get_pk_inc=0;
unsigned long long ari_get_pk_call_total=0;

static unsigned long ari_s_hash[387] = {
0x00000200,0x00000B01,0x00000C02,0x00000D03,0x00000F25,0x0000101C,0x0000110B,
0x00001327,
0x0000142F,0x00002B25,0x00002C22,0x00002D14,0x00002F2D,0x0000302B,0x0000312B,
0x00003330,
0x00003432,0x00003532,0x00004C32,0x00005031,0x00005131,0x0000FA02,0x0000FB01,
0x0000FC1C,
0x0000FE1C,0x0000FF1C,0x0001001E,0x00010A2E,0x00010B25,0x00010E25,0x00010F25,
0x00013938,
0x00013A04,0x00013B02,0x00013C01,0x0010393D,0x00107504,0x00107605,0x00107706,
0x0010790D,
0x00107A07,0x00107B08,0x0010850D,0x00108609,0x0010870A,0x00108B0E,0x0010B50B,
0x0010B60C,
0x0010B70D,0x0010B90B,0x0010BA1D,0x0010BB16,0x0010C615,0x0010C70C,0x0010F521,
0x0010F628,
0x0010F728,0x00110528,0x00117516,0x0011760E,0x0011770F,0x00117A12,0x00117B07,
0x0011870E,
0x0011B514,0x0011B615,0x0011B70C,0x0011B914,0x0011BA15,0x0011BB1D,0x0011C619,
0x0011C715,
0x00147516,0x00147610,0x00147711,0x0014791D,0x00147A0C,0x00147B0E,0x0014851B,
0x00148616,
0x00148707,0x0014890B,0x00148A1D,0x00148B16,0x0014950B,0x0014961D,0x0014B615,
0x0014B71D,
0x0014BA14,0x0014BB15,0x0014C614,0x0014C715,0x0015052D,0x0015750B,0x0015760C,
0x00157710,
0x0015790B,0x00157A1D,0x00157B16,0x0015850B,0x0015861D,0x0015870C,0x00158914,
0x00158A15,
0x00158B1D,0x0015B619,0x0015B714,0x0020751D,0x00207612,0x00207713,0x0020791D,
0x00207A0C,
0x00207B0E,0x0020850B,0x0020860C,0x00208710,0x00208A1D,0x00208B0C,0x0020B615,
0x0020B71D,
0x0021750B,0x0021760C,0x0021770E,0x0021790B,0x00217A1D,0x00217B12,0x00218514,
0x00218615,
0x0021870C,0x0021891C,0x00218A15,0x00218B1D,0x0021B619,0x0021B715,0x0021BA22,
0x0021BB19,
0x00247514,0x0024761D,0x0024770E,0x00247914,0x00247A15,0x00247B0C,0x00248514,
0x0024861D,
0x00248716,0x0024891C,0x00248A15,0x00248B1D,0x0024B619,0x0024B715,0x0025751C,
0x0025760B,
```

도면17b

```
0x0025771B,0x0025791C,0x00257A0B,0x00257B1B,0x0025851C,0x0025860B,0x0025871B,
0x0025890B,
0x00258A1D,0x00258B1B,0x0025B618,0x0025B722,0x0025BA1F,0x0025BB18,0x0025C61F,
0x0025C718,
0x0025CA1F,0x0025CB1F,0x003A9D1C,0x003A9E0B,0x003A9F0B,0x004FB125,0x004FB21C,
0x004FB31C,
0x00907514,0x00907615,0x00907716,0x00907919,0x00907A15,0x00907B1D,0x00907D1C,
0x00907E1C,
0x00907F14,0x00908522,0x00908614,0x0090871D,0x00908918,0x00908A19,0x00908B14,
0x00908D24,
0x00909523,0x0090961F,0x0090992B,0x0090B517,0x0090B618,0x0090B719,0x0090B91F,
0x0090BA22,
0x0090BB19,0x0090BD1C,0x0090BE1C,0x0090BF1C,0x0090C52B,0x0090C61F,0x0090C718,
0x0090C917,
0x0090CA1F,0x0090CB1F,0x0090CD23,0x0090D52E,0x0090D62C,0x0090D92C,0x0090F52D,
0x0090F62D,
0x0090F72F,0x0090F925,0x0090FA2E,0x0090FB2D,0x0090FD1E,0x0090FE1E,0x0091052F,
0x0091062F,
0x00910928,0x00910D25,0x00917519,0x00917615,0x0091771D,0x00917922,0x00917A14,
0x00917B15,
0x00918619,0x00918714,0x00918A18,0x00918B18,0x0091B618,0x0091B722,0x0091BA18,
0x0091BB18,
0x00947518,0x00947614,0x0094771D,0x0094791F,0x00947A19,0x00947B14,0x00948520,
0x00948619,
0x00948714,0x00948A18,0x00948B18,0x0094B61F,0x0094B718,0x0094BA17,0x0094BB1F,
0x0094C617,
0x0094C717,0x00957520,0x00957622,0x00957714,0x00957924,0x00957A18,0x00957B18,
0x00958524,
0x00958618,0x00958718,0x0095892B,0x00958A17,0x00958B1F,0x0095B52B,0x0095B617,
0x0095B71F,
0x0095B92B,0x0095BA17,0x0095BB17,0x0095C52C,0x0095C617,0x0095C717,0x0095C92C,
0x0095CA2B,
0x0095CB2C,0x00A0751F,0x00A07614,0x00A07715,0x00A0791F,0x00A07A19,0x00A07B14,
0x00A0851F,
0x00A08622,0x00A08714,0x00A08917,0x00A08A18,0x00A08B18,0x00A0B52B,0x00A0B61F,
0x00A0B718,
0x00A0BA17,0x00A0BB1F,0x00A0C617,0x00A0C717,0x00A17524,0x00A17622,0x00A17714,
0x00A17924,
0x00A17A18,0x00A17B18,0x00A1861F,0x00A18718,0x00A18A17,0x00A18B17,0x00A1B617,
0x00A1B71F,
0x00A1BA17,0x00A1BB17,0x00A47524,0x00A47618,0x00A47714,0x00A4792B,0x00A47A18,
0x00A47B18,
0x00A4852B,0x00A4861F,0x00A48718,0x00A4892B,0x00A48A17,0x00A48B17,0x00A4B52C,
0x00A4B617,
0x00A4B717,0x00A4B92C,0x00A4BA17,0x00A4BB17,0x00A4C52E,0x00A4C62B,0x00A4C72B,
0x00A4C92E,
0x00A4CA2C,0x00A4CB2C,0x00A5752C,0x00A57617,0x00A57718,0x00A5792B,0x00A57A17,
0x00A57B1F,
0x00A5852C,0x00A58617,0x00A5871F,0x00A5892C,0x00A58A17,0x00A58B17,0x00A5B52E,
0x00A5B62B,
0x00A5B717,0x00A5B92E,0x00A5BA2C,0x00A5BB2C,0x00A5C52E,0x00A5C62C,0x00A5C72C,
0x00A5C92E,
0x00A5CA2C,0x00A5CB2C,0x00BA9D2D,0x00BA9E2E,0x00BA9F2E,0x00CDD938,0x00CE2D38,
0x00CEE938,
0x00CF2D38,0x00D02D38,0x00D0313A,0x00D0713A,0x0110762F,0x0110B632,0x0110B731,
0x03D0113B,
0x03D0213C,0x03D02D3D,0x03D0313D,0x03D0413C,0x03D04D3D,0x03D0513C,0x03D05D3D,
0x03D06139,
0x03D0693D,0x03D06D3D,0x03D0713D
};
```

도면18

```
static unsigned long ari_gs_hash[225] = {
0x00000401,0x0001491A,0x0001590B,0x00017621,0x0001891C,0x0009492A,0x000DFA38,
0x000F6D3D,
0x0010003B,0x0010B21B,0x0011321C,0x00116B29,0x0011B31D,0x00126B1E,0x00136623,
0x00146729,
0x00146F3B,0x0015321F,0x00156E27,0x00163320,0x00182725,0x00186727,0x00196323,
0x001C4721,
0x001E3F30,0x001E433B,0x00203F2A,0x0020463B,0x0020F322,0x00216A2E,0x00226723,
0x00245625,
0x00256724,0x00286625,0x002D3726,0x002D573A,0x00316627,0x00326628,0x00344729,
0x00366628,
0x003D4329,0x00416A2A,0x0042533A,0x00916A2A,0x00926B2B,0x0093E72E,0x00956B2C,
0x009D362D,
0x009D3B39,0x009E4330,0x00A2672E,0x00AD372F,0x01145630,0x01146B27,0x011C8231,
0x01226732,
0x012CC333,0x01413B34,0x019CA335,0x019CB338,0x01ACB736,0x01AD823D,0x01C37F37,
0x02156738,
0x0218AB3B,0x021C9B35,0x021E0738,0x021FB73D,0x0220E335,0x02216B3C,0x02217234,
0x0222B33C,
0x02239B3B,0x0223B23A,0x0224673B,0x0238A739,0x0240B23D,0x024CBF38,0x024CC23D,
0x024D8738,
0x0297AF3A,0x02986727,0x0298A33B,0x0298A738,0x029CAF3B,0x029CC33A,0x02A0AB35,
0x02A3E736,
0x02AC773B,0x02B0B335,0x02B3A73B,0x02C0D73C,0x02C1E735,0x03108E3D,0x03109737,
0x0311D639,
0x03147F3C,0x0314B236,0x0317A639,0x0317D629,0x0317DB33,0x03187627,0x0318AF3B,
0x0318F61A,
0x0319D739,0x031C953B,0x031D633C,0x031FCF39,0x0320873B,0x0320963A,0x03222639,
0x0323833C,
0x03239A27,0x0323EA2F,0x03242631,0x03242B3B,0x03249727,0x0325AB39,0x0327A73C,
0x0327C728,
0x03287727,0x03287E3A,0x03288737,0x032BAA39,0x032C7527,0x032D2337,0x032E9B39,
0x032EA23B,
0x032EBF3C,0x032F7E39,0x0330C63C,0x0332B23B,0x0332F230,0x03339F3B,0x0333EE27,
0x03348F30,
0x0336AB3C,0x0338A73B,0x033A7639,0x033A7F1A,0x033C793B,0x033C9A34,0x033CA33B,
0x033CA738,
0x033D0A3C,0x033DB339,0x033DFF3C,0x033E9739,0x0340CB3C,0x0344573B,0x0344AA3C,
0x0348263B,
0x034C7B3C,0x034CBB3A,0x034CD33C,0x0390B73D,0x0390E937,0x0393653D,0x0394B73B,
0x0394E33D,
0x0394FA38,0x03950A3C,0x0396CF3D,0x03971A36,0x0398673C,0x0398E13B,0x03994E39,
0x039C733B,
0x039D191A,0x039D4536,0x039E053C,0x039E6E3D,0x039E9D34,0x039F8D39,0x03A0C93B,
0x03A67939,
0x03A69D29,0x03A6D637,0x03A85A3C,0x03AE5B3B,0x03AEDB3D,0x03AF2E3C,0x03B0A13B,
0x03B2B139,
0x03B3123B,0x03B36339,0x03B3AD3C,0x03B42E33,0x03B4733B,0x03B4F53C,0x03B51F36,
0x03B59139,
0x03B5CB3C,0x03B61737,0x03B93A3C,0x03B98F39,0x03B9F53C,0x03BA063B,0x03BA2A3C,
0x03BB2739,
0x03BD3B3B,0x03BDC939,0x03BDF534,0x03BF9A39,0x03C1653B,0x03C19E2A,0x03C20527,
0x03C3633B,
0x03C3823C,0x03C3A527,0x03C45A3B,0x03C4993C,0x03C5B23B,0x03C5D527,0x03C9563B,
0x03C9A93C,
0x03CA063B,0x03CB0E3C,0x03CCB53B,0x03CD1E3C,0x03CED23D,0x03CEDF3C,0x03CFFA39,
0x40BC673E,
0xFFFFFF3F
};
```

도면19a

```
unsigned short ari_cf_m[64][9] = {
{65535,65534,65532,65215,  321,    4,    2,    1,    0},   ← pki=0
{65490,65339,64638,58133, 7463,  973,  270,  125,    0},   ← pki=1
{65530,65509,65319,60216, 5308,  222,   30,    9,    0},
{65534,65528,65470,62535, 3012,   67,    8,    2,    0},
{65533,65524,65435,62110, 3434,  104,   14,    5,    0},
{65535,65533,65499,62363, 3173,   37,    3,    1,    0},
{65535,65534,65522,63164, 2371,   14,    2,    1,    0},
{65535,65530,65448,59939, 5612,   88,    7,    2,    0},
{65535,65533,65500,61498, 4044,   38,    3,    1,    0},
{65535,65530,65444,59855, 5667,   92,    6,    1,    0},
{65535,65532,65495,61386, 4140,   39,    3,    1,    0},
{65522,65458,64905,55424,10056,  634,   88,   28,    0},
{65532,65511,65238,57072, 8457,  297,   27,    6,    0},
{65534,65522,65364,59096, 6461,  171,   15,    3,    0},
{65535,65530,65426,59204, 6342,  109,    8,    2,    0},
{65535,65533,65492,61008, 4512,   43,    3,    1,    0},
{65535,65529,65417,58998, 6519,  118,    6,    1,    0},
{65535,65533,65490,60856, 4679,   46,    4,    1,    0},
{65535,65528,65384,58400, 7127,  149,    9,    1,    0},
{65535,65532,65483,60544, 4984,   56,    4,    1,    0},
{65517,65413,64537,53269,12264, 1002,  138,   38,    0},
{65531,65503,65125,55553, 9985,  420,   37,    7,    0},
{65534,65518,65303,57889, 7650,  235,   20,    3,    0},
{65490,65288,63679,49500,15949, 1903,  301,   94,    0},
{65522,65428,64429,51580,13957, 1113,  114,   22,    0},
{65526,65447,64600,52808,12743,  937,   93,   17,    0},
{63814,60228,53108,40709,26294,15412, 8961, 5729,    0},
{65526,65486,65133,57227, 8244,  400,   58,   20,    0},
{65500,65346,64297,52845,12477, 1283,  230,   70,    0},
{65528,65486,65077,56652, 8871,  465,   56,   16,    0},
{65464,65186,63581,50731,14351, 1992,  396,  128,    0},
{65489,65278,63861,51225,14185, 1726,  302,   96,    0},
{65485,65249,63632,50425,14933, 1943,  332,   96,    0},
{65292,64495,61270,47805,17600, 4502, 1337,  542,    0},
{65519,65421,64478,52517,12971, 1068,  129,   33,    0},
{65470,65181,63344,49862,15299, 2233,  418,  132,    0},
{65472,65197,63407,49933,15445, 2176,  396,  123,    0},
```

도면19b

```
{65376,64781,62057,48496,16676, 3614,  923,  340,    0},
{65259,64356,60836,47316,18158, 4979, 1517,  623,    0},
{64883,63190,58260,45006,21034, 8378, 3559, 1909,    0},
{65261,64180,60126,46710,18694, 5578, 1582,  531,    0},
{64933,63355,58991,46299,19470, 7245, 2989, 1449,    0},
{63999,61383,56309,44712,24964,14237, 9489, 7028,    0},
{65451,65091,62953,48747,16324, 2626,  522,  168,    0},
{65400,64870,62109,47037,18198, 3526,  794,  278,    0},
{65200,64074,59673,44322,20692, 6133, 1836,  739,    0},
{65376,64798,61822,46437,18673, 3881,  932,  368,    0},
{65151,63887,59083,43617,21491, 6768, 2081,  841,    0},
{64592,62314,56211,42184,24450,11142, 5265, 3075,    0},
{64908,62840,56205,41474,23652, 9844, 3388, 1379,    0},
{65021,63308,57341,42286,22972, 8709, 2895, 1232,    0},
{64790,62474,55461,40843,24327,10719, 3921, 1677,    0},
{64053,60476,52429,39583,26962,15208, 7592, 4166,    0},
{63317,58934,51305,40469,29263,19682,12661, 8553,    0},
{63871,59872,52031,39473,26093,15132, 7866, 4080,    0},
{63226,58553,50425,39191,28586,18779,11388, 7035,    0},
{62219,57006,49569,40492,32376,24784,18716,14447,    0},
{62905,58273,50651,39619,28123,18379,11633, 7478,    0},
{63420,59073,51922,41516,29863,20328,13529, 9237,    0},
{63582,59263,51165,37880,24026,13893, 7771, 4535,    0},
{63223,58418,49833,37279,25503,15421, 9122, 5802,    0},
{62322,56878,48746,39095,30723,22195,15849,11887,    0},
{61826,47222,47123,47015,46913,46806,13713, 6895,    0},
{60678,44085,44084,44083,44082,44081,16715, 9222,    0}   ← pki=63
};
```

도면20a

```
static unsigned long ari_s_hash[387] = {
0x0090D52E,0x0090CB1F,0x00A4CB2C,0x00003330,0x00107A07,0x00907A15,0x00207A0C,
0x00147A0C,
0x00247A15,0x00A07A19,0x00947A19,0x00A47A18,0x0010B70D,0x0090B719,0x0020B71D,
0x0014B71D,
0x00A0B718,0x0094B718,0x0024B715,0x00A4B717,0x0110B731,0x0000FE1C,0x0090FE1E,
0x00013B02,
0x00A5C92E,0x0095C92C,0x003A9E0B,0x00000B01,0x00BA9E2E,0x0090992B,0x0011B514,
0x00A5B52E,
0x0095B52B,0x0090D62C,0x0010850D,0x0014851B,0x00248514,0x0020850B,0x00908522,
0x00948520,
0x00A4852B,0x00A0851F,0x00003432,0x00107B08,0x00207B0E,0x00907B1D,0x00147B0E,
0x00247B0C,
0x00A07B14,0x00947B14,0x00A47B18,0x00910928,0x03D0713D,0x00D0713A,0x0000FF1C,
0x0090F52D,
0x0010F521,0x00013C01,0x03D05D3D,0x00A5CA2C,0x0025CA1F,0x0095CA2B,0x003A9F0B,
0x00000C02,
0x0021790B,0x0025791C,0x00917922,0x0015790B,0x00A17924,0x00A5792B,0x00957924,
0x00BA9F2E,
0x00CF2D38,0x00000200,0x0011B615,0x0091B618,0x0021B619,0x0015B619,0x00A1B617,
0x0095B617,
0x0025B618,0x00A5B62B,0x004FB125,0x00108609,0x00148616,0x0020860C,0x00908614,
0x0024861D,
0x00948619,0x00A08622,0x00A4861F,0x0090CD23,0x00003532,0x00010A2E,0x00002B25,
0x0010B90B,
0x0090B91F,0x00A4B92C,0x0001001E,0x03D0213C,0x0090F62D,0x0010F628,0x00A5CB2C,
0x0025CB1F,
0x0095CB2C,0x00000D03,0x00117A12,0x00217A1D,0x00917A14,0x00257A0B,0x00157A1D,
0x00A17A18,
0x00A57A17,0x00957A18,0x0011B70C,0x0091B722,0x0021B715,0x0015B714,0x00A1B71F,
0x0025B722,
0x0095B71F,0x00A5B717,0x004FB21C,0x0010870A,0x00148707,0x00208710,0x0090871D,
0x00248716,
0x00948714,0x00A08714,0x00A48718,0x00907D1C,0x00010B25,0x00002C22,0x0010BA1D,
0x0090BA22,
0x0014BA14,0x00A0BA17,0x0094BA17,0x00A4BA17,0x03D0693D,0x0090F72F,0x0010F728,
0x0025851C,
0x0015850B,0x00218514,0x00A5852C,0x00958524,0x00117B07,0x00217B12,0x00257B1B,
0x00917B15,
0x00157B16,0x00A17B18,0x00A57B1F,0x00957B18,0x0090D92C,0x004FB31C,0x03D0413C,
0x00907E1C,
0x0090C52B,0x00A4C52E,0x00002D14,0x00D02D38,0x03D02D3D,0x0010BB16,0x0090BB19,
0x0014BB15,
0x00A0BB1F,0x0094BB1F,0x00A4BB17,0x0025860B,0x0015861D,0x00218615,0x00918619,
0x00A58617,
0x00958618,0x00A1861F,0x00000F25,0x00CEE938,0x00004C32,0x0011B914,0x00A5B92E,
0x0095B92B,
0x0014890B,0x0024891C,0x00908918,0x00A4892B,0x00A08917,0x00907F14,0x0010C615,
0x0090C61F,
0x0014C614,0x0094C617,0x00A4C62B,0x00A0C617,0x00910D25,0x00107504,0x00907514,
0x00147516,
0x0020751D,0x00247514,0x00A0751F,0x00947518,0x00A47524,0x0090F925,0x0011870E,
0x0025871B,
0x0015870C,0x0021870C,0x00918714,0x00A5871F,0x00A18718,0x00958718,0x03D06139,
0x0000101C,
0x03D04D3D,0x0011BA15,0x0091BA18,0x0021BA22,0x00A1BA17,0x0025BA1F,0x00A5BA2C,
0x0095BA17,
0x00148A1D,0x00208A1D,0x00248A15,0x00908A19,0x00948A18,0x00A08A18,0x00A48A17,
0x0010393D,
```

도면20b

```
0x0010C70C,0x0090C718,0x0014C715,0x0094C717,0x00A0C717,0x00A4C72B,0x00010E25,
0x00002F2D,
0x00107605,0x00907615,0x00147610,0x00207612,0x0024761D,0x00A07614,0x00947614,
0x00A47618,
0x0110762F,0x0090BD1C,0x00CDD938,0x0000FA02,0x0090FA2E,0x0000110B,0x03D0113B,
0x00A5C52E,
0x0095C52C,0x0011BB1D,0x0091BB18,0x0021BB19,0x0014950B,0x00A1BB17,0x0025BB18,
0x00A5BB2C,
0x0095BB17,0x00909523,0x00108B0E,0x00148B16,0x00208B0C,0x00248B1D,0x00908B14,
0x00948B18,
0x00A08B18,0x00A48B17,0x00010F25,0x0000302B,0x00107706,0x00907716,0x00147711,
0x00207713,
0x0024770E,0x00A07715,0x0094771D,0x00A47714,0x0091052F,0x00110528,0x0090BE1C,
0x0015052D,
0x03D06D3D,0x0000FB01,0x0090FB2D,0x0025890B,0x00A5892C,0x00158914,0x0021891C,
0x0095892B,
0x0011C619,0x0025C61F,0x00A5C62C,0x0095C617,0x00117516,0x0021750B,0x0015750B,
0x00917519,
0x0025751C,0x00A17524,0x00957520,0x00A5752C,0x0014961D,0x0090961F,0x0090C917,
0x00A4C92E,
0x0000312B,0x00D0313A,0x03D0313D,0x0090BF1C,0x0091062F,0x0010B50B,0x0090B517,
0x00A0B52B,
0x00A4B52C,0x0000FC1C,0x00258A1D,0x00A58A17,0x00158A15,0x00218A15,0x00918A18,
0x00A18A17,
0x00958A17,0x00013938,0x00001327,0x0011C715,0x0025C718,0x00A5C72C,0x0095C717,
0x0011760E,
0x0021760C,0x00917615,0x0015760C,0x0025760B,0x00A17622,0x00957622,0x00A57617,
0x00005031,
0x00908D24,0x0090CA1F,0x00A4CA2C,0x0010790D,0x00907919,0x0020791D,0x0014791D,
0x00247914,
0x00A0791F,0x0094791F,0x00A4792B,0x00CE2D38,0x0010B60C,0x0090B618,0x0014B615,
0x0020B615,
0x00A0B61F,0x0094B61F,0x0024B619,0x00A4B617,0x0110B632,0x00258B1B,0x00158B1D,
0x00218B1D,
0x00A58B17,0x00918B18,0x00A18B17,0x00958B1F,0x0090FD1E,0x00013A04,0x0000142F,
0x003A9D1C,
0x00BA9D2D,0x0011770F,0x0021770E,0x00157710,0x0091771D,0x0025771B,0x00A17714,
0x00957714,
0x00A57718,0x00005131,0x03D0513C
};
```