(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2014년07월11일
(11) 등록번호 10-1419148
(24) 등록일자 2014년07월07일

(51) 국제특허분류(Int. Cl.)
G10L 19/00 (2006.01)
(21) 출원번호 10-2012-7012640
(22) 출원일자(국제) 2010년10월19일
심사청구일자 2012년05월16일
(85) 번역문제출일자 2012년05월16일
(65) 공개번호 10-2012-0074306
(43) 공개일자 2012년07월05일
(86) 국제출원번호 PCT/EP2010/065727
(87) 국제공개번호 WO 2011/048100
국제공개일자 2011년04월28일
(30) 우선권주장
61/253,459 2009년10월20일 미국(US)
(56) 선행기술조사문헌
NEUENDORF MAX ET AL, " A Novel Scheme for Low Bitrate Unified Speech and Audio Coding MEPG RMO", MAY 2009
전체 청구항 수 : 총 14 항

(73) 특허권자
프라운호퍼 게젤샤프트 쭈르 푀르데룽 데어 안겐반텐 포르슝 에. 베.
독일 80686 뮌헨 한자슈트라쎄 27 체
(72) 발명자
푸쉬 귈라움
독일 에를랑겐 91058 퓌르터 스트라세 17
수바라만 비네쉬
독일 게르메링 82110 루드비히 스트라쎄 16
(뒷면에 계속)
(74) 대리인
특허법인 정안

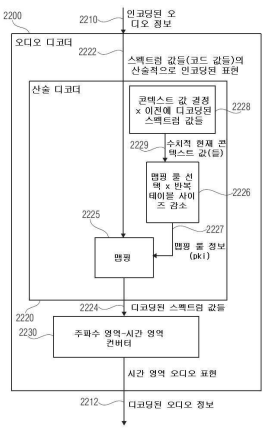
심사관 : 이수철

(54) 발명의 명칭 **반복 구간 사이즈 감소를 이용한 오디오 인코더, 오디오 디코더, 오디오 정보 인코딩 방법, 오디오 정보 디코딩 방법, 및 컴퓨터 프로그램**

(57) 요 약

인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더(2200)는 스펙트럼 계수들의 산술적으로 인코딩된 표현(2222)에 기초하여 복수의 디코딩된 스펙트럼 값들(2224)을 제공하기 위한 산술 디코더(2200)를 포함한다. 오디오 디코더는 또한 디코딩된 오디오 정보(2212)를 획득하기 위해, 디코딩된 스펙트럼 값들(2224)을 이용하여 시간 영역 오디오 표현을 제공하기 위한 주파수 영역-대-시간 영역 컨버터(2230)를 포함한다. 산술 디코더는 현재 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값에 의존하여 심볼 코드로의 코드 값의 맵핑을 기술하는 맵핑 룰을 선택하도록 구성된다. 산술 디코더는 이전에 디코딩된 복수의 스펙트럼 값들에 의존하여 수치적 현재 콘텍스트 값을 결정하도록 구성된다. 산술 디코더는 반복 구간 사이즈 감소를 이용하여 적어도 하나의 테이블을 평가하고, 수치적 현재 콘텍스트 값이 테이블의 엔트리에 의해 기술된 테이블 콘텍스트 값과 동일하거나 또는 테이블의 엔트리들에 의해 기술된 구간 내에 놓여 있는지 여부를 결정하며, 선택된 맵핑 테이블을 기술하는 맵핑 룰 인덱스 값을 유도하도록 구성된다. 오디오 인코더는 또한 반복 구간 테이블 사이즈 감소를 이용한다.

대 표 도 - 도22



(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 제공하기 위한 오디오 디코더(200; 800; 2200)에 있어서,

스펙트럼 값들의 산술적으로 인코딩된 표현(222; 821; 2222)에 기초하여 복수의 디코딩된 스펙트럼 값들(232; 822; 2224)을 제공하기 위한 산술 디코더(230; 820; 2220); 및

상기 디코딩된 오디오 정보(212; 812; 2212)를 획득하기 위해, 상기 디코딩된 스펙트럼 값들(232; 822; 2224)을 이용하여 시간 영역 오디오 표현(262; 812; 2212)을 제공하기 위한 주파수 영역-대-시간 영역(frequency-domain-to-time-domain) 컨버터(260; 830; 2230)

를 포함하며,

상기 산술 디코더(230; 820; 2220)는 현재 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값(numeric current context value)(s)에 의존하여 심볼 코드(symbol)로의 코드 값(value)의 맵핑을 기술하는 맵핑 룰(297; cum_freq[])을 선택하도록 구성되고,

상기 산술 디코더는 이전에 디코딩된 복수의 스펙트럼 값들(a)에 의존하여 상기 수치적 현재 콘텍스트 값(s)을 결정하도록 구성되고,

상기 산술 디코더는 반복 구간(iterative interval) 사이즈 감소(542; 546)를 이용하여 적어도 하나의 테이블(ari_s_hash[387]; ari_gs_hash[225])을 평가하고, 상기 수치적 현재 콘텍스트 값(s)이 상기 테이블의 엔트리(j, ari_s_hash[i], ari_gs_hash[i])에 의해 기술된 테이블 콘텍스트 값과 동일하거나 또는 상기 테이블의 엔트리들에 의해 기술된 구간 내에 놓여 있는지 여부를 결정하며, 선택된 맵핑 룰(arith_cf_m[pki][9])을 기술하는 맵핑 룰 인덱스 값(pki)을 유도하도록 구성되는, 오디오 디코더(200; 800).

청구항 2

제1항에 있어서, 상기 산술 디코더(230; 820)는,

초기 테이블 구간의 하위 경계를 지정하기 위해 하위 구간 경계 변수(i_min)를 초기화하고,

상기 초기 테이블 구간의 상위 경계를 지정하기 위해 상위 구간 경계 변수(i_max)를 초기화하고,

상기 초기 테이블 구간의 중심에 테이블 인덱스(i)가 배열되어 있는 테이블 엔트리(ari_s_hash[i], ari_gs_hash[i])를 평가하고, 이 평가된 테이블 엔트리(ari_s_hash[i], ari_gs_hash[i])에 의해 표현된 테이블 콘텍스트 값(j >> 8)과 상기 수치적 현재 콘텍스트 값(s)을 비교하고,

업데이트된 테이블 구간을 획득하기 위해, 상기 비교의 결과에 의존하여 상기 하위 구간 경계 변수(i_min) 또는 상기 상위 구간 경계 변수(i_max)를 조정하며,

테이블 콘텍스트 값이 상기 수치적 현재 콘텍스트 값(s)과 동일하거나 또는 업데이트된 구간 경계 변수들(i_min, i_max)에 의해 정의된 테이블 구간의 사이즈가 테이블 구간 사이즈 문턱값(threshold)에 도달하거나 또는 그 아래로 내려갈 때 까지, 하나 이상의 업데이트된 테이블 구간들에 기초하여 테이블 엔트리의 평가 및 상기 하위 구간 경계 변수 또는 상기 상위 구간 경계 변수의 조정을 반복하도록 구성되는, 오디오 디코더(200; 800).

청구항 3

제2항에 있어서, 상기 산술 디코더(230; 820)는 테이블(ari_s_hash, ari_gs_hash)의 주어진 엔트리가 상기 수치적 현재 콘텍스트 값(s)과 동일한 테이블 콘텍스트 값(j >> 8)을 표현한다는 것을 발견한 것에 응답하여 상기 테이블의 상기 주어진 엔트리(ari_s_hash[i], ari_gs_hash[i])에 의해 기술된 맵핑 룰 인덱스 값(pki)을 제공하도록 구성되는, 오디오 디코더(200; 800).

청구항 4

제1항에 있어서, 상기 산술 디코더(230; 820)는,

a) 하위 구간 경계 변수 i_min을 -1로 설정하고;

b) 상위 구간 경계 변수 i_max를 테이블 엔트리들의 갯수에서 1을 뺀 수로 설정하고;

c) i_max와 i_min간의 차이가 1보다 큰지 여부를 체크하고, 이러한 조건이 더 이상 충족되지 않거나 또는 중지 조건에 도달할 때 까지 아래의 단계들을 반복하는 알고리즘을 수행하도록 구성되며, 상기 아래의 단계들은,

c1) 변수 i를 i_min + ((i_max - i_min)/2)로 설정하는 단계,

c2) 테이블 인덱스 i 를 갖는 테이블 엔트리에 의해 기술된 테이블 콘텍스트 값이 상기 수치적 현재 콘텍스트 값보다 큰 경우 상위 구간 경계 변수 i_max를 i로 설정하고, 테이블 인덱스 i 를 갖는 테이블 엔트리에 의해 기술된 테이블 콘텍스트 값이 상기 수치적 현재 콘텍스트 값보다 작은 경우 하위 구간 경계 변수 i_min을 i로 설정하는 단계, 및

c3) 테이블 인덱스 i 를 갖는 테이블 엔트리에 의해 기술된 테이블 콘텍스트 값이 상기 수치적 현재 콘텍스트 값과 동일한 경우, 상기 (c)의 반복을 중지하고, 테이블 인덱스 i 를 갖는 상기 테이블 엔트리에 의해 기술된 맵핑 룰 인덱스 값(pki)을 상기 알고리즘의 결과로서 반환하는 단계를 포함하는, 오디오 디코더(200; 800).

청구항 5

제1항에 있어서, 상기 산술 디코더는 이전에 디코딩된 스펙트럼 값들(a)의 크기들을 기술하는 크기 값들(c0, c1, c2, c3, c4, c5, c6)의 가중화된 조합에 기초하여 상기 수치적 현재 콘텍스트 값(s)을 획득하도록 구성되는, 오디오 디코더(200; 800).

청구항 6

제1항에 있어서, 상기 테이블(ari_s_hash, ari_gs_hash)은 복수의 엔트리들을 포함하며,

상기 복수의 엔트리들 각각은 테이블 콘텍스트 값(j >> 8) 및 연관된 맵핑 룰 인덱스 값(j& 0xFF, pki)을 기술하며,

상기 테이블의 엔트리들은 상기 테이블 콘텍스트 값들에 따라 수치적으로 순서화되는, 오디오 디코더(200; 800).

청구항 7

제1항에 있어서, 상기 테이블은 복수의 엔트리들을 포함하며,

복수의 엔트리들 각각은 콘텍스트 값 구간의 경계 값을 정의하는 테이블 콘텍스트 값과, 상기 콘텍스트 값 구간과 연관된 맵핑 룰 인덱스 값(pki)을 기술하는, 오디오 디코더(200; 800).

청구항 8

제1항에 있어서, 상기 산술 디코더(230; 820)는 상기 수치적 현재 콘텍스트 값(s)에 의존하여 두 단계의 맵핑 룰 선택을 수행하도록 구성되고,

상기 산술 디코더는, 제1 선택 단계(540)에서, 상기 수치적 현재 콘텍스트 값(s), 또는 이로부터 유도된 값이 다이렉트 히트 테이블(ari_s_hash)의 엔트리(j, ari_s_hash[i])에 의해 기술된 중요 상태 값(significant state value)(j >> 8)과 동일한지 여부를 체크하도록 구성되고,

상기 산술 디코더는 상기 수치적 현재 콘텍스트 값(s), 또는 이로부터 유도된 값이 상기 다이렉트 히트 테이블(ari_s_hash)의 엔트리들에 의해 기술된 중요 상태 값들과 상이한 경우에만 실행되는 제2 선택 단계(544)에서, 복수의 구간들 중어느 구간에 상기 수치적 현재 콘텍스트 값(s)이 놓여 있는지를 결정하도록 구성되며,

상기 산술 디코더는 반복 구간 사이즈 감소(542)를 이용하여 상기 다이렉트 히트 테이블(ari_s_hash)을 평가하고, 상기 수치적 현재 콘텍스트 값(s)이 상기 다이렉트 히트 테이블(ari_s_hash)의 엔트리(ari_s_hash[i])에 의해 기술된 테이블 콘텍스트 값(j >> 8)과 동일한지 여부를 결정하도록 구성되는, 오디오 디코더(200; 800).

청구항 9

제8항에 있어서, 상기 산술 디코더는, 상기 제2 선택 단계(544)에서, 구간 맵핑 테이블(ari_gs_hash)을 평가하도록 구성되며, 이 테이블의 엔트리들은 반복 구간 사이즈 감소(546)를 이용하여 콘텍스트 값 구간들의 경계 값들을 기술하는, 오디오 디코더(200; 800).

청구항 10

제9항에 있어서, 상기 산술 디코더(230; 820)는, 테이블 구간의 사이즈가 미리결정된 테이블 구간 사이즈 문턱값에 도달하거나 또는 그 아래로 감소하거나, 또는 상기 테이블 구간의 중심에서 테이블 엔트리(j, ari_gs_hash[i])에 의해 기술된 상기 구간 경계 콘텍스트 값이 상기 수치적 현재 콘텍스트 값(s)과 동일할 때까지, 엔트리들(ari_gs_hash[i])에 의해 표현된 구간 경계 콘텍스트 값들(j >> 8)과 상기 수치적 현재 콘텍스트 값(s) 간의 비교에 의존하여 상기 테이블 구간의 사이즈를 반복적으로 감소시키도록 구성되며,

상기 산술 디코더는 상기 테이블 구간의 사이즈의 반복적인 감소가 중지될 때 상기 테이블 구간의 구간 경계의 설정에 의존하여 상기 맵핑 룰 인덱스 값(pki)을 제공하도록 구성되는, 오디오 디코더(200; 800).

청구항 11

입력 오디오 정보(110;710;2110)에 기초하여 인코딩된 오디오 정보(112;712; 2112)를 제공하기 위한 오디오 인코더(100; 700; 2100)에 있어서,

주파수 영역 오디오 표현(132;722;2124)이 스펙트럼 값들의 세트를 포함하도록, 상기 입력 오디오 정보의 시간 영역 표현에 기초하여 주파수 영역 오디오 표현을 제공하기 위한 에너지 압축 시간 영역-대-주파수 영역 컨버터(130;720;2120); 및

가변 길이 코드워드(acod_m, acod_r)를 이용하여 스펙트럼 값(a) 또는 이것의 사전처리된 버전을 인코딩하도록 구성된 산술 인코더(170;730;2130)

를 포함하며,

상기 산술 인코더(170)는 스펙트럼 값(a), 또는 스펙트럼 값(a)의 최상위 비트플레인(most-significant bitplane)의 값(m)을 코드 값(acod_m)에 맵핑하도록 구성되고,

상기 산술 인코더는 현재 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값(s)에 의존하여 코드 값으로의 스펙트럼 값, 또는 스펙트럼 값의 최상위 비트플레인의 맵핑을 기술하는 맵핑 룰을 선택하도록 구성되고,

상기 산술 인코더는 이전에 인코딩된 복수의 스펙트럼 값들에 의존하여 상기 수치적 현재 콘텍스트 값(s)을 결정하도록 구성되며,

상기 산술 인코더는 반복 구간(iterative interval) 사이즈 감소를 이용하여 적어도 하나의 테이블(ari_s_hash, ari_gs_hash)을 평가하고, 상기 수치적 현재 콘텍스트 값(s)이 테이블의 엔트리(ari_s_hash[i], ari_gs_hash[i])에 의해 기술된 콘텍스트 값과 동일하거나 또는 상기 테이블의 엔트리들에 의해 기술된 구간 내에 놓여 있는지 여부를 결정하며, 선택된 맵핑 룰을 기술하는 맵핑 룰 인덱스 값(pki)을 유도하도록 구성되는, 오디오 인코더(100; 700; 2100).

청구항 12

인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 방법에 있어서,

스펙트럼 값들의 산술적으로 인코딩된 표현에 기초하여 복수의 디코딩된 스펙트럼 값들을 제공하는 단계; 및

디코딩된 오디오 정보를 획득하기 위해, 상기 디코딩된 스펙트럼 값들을 이용하여 시간 영역 오디오 표현을 제공하는 단계

를 포함하고,

상기 복수의 디코딩된 스펙트럼 값들을 제공하는 단계는, 현재 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값(s)에 의존하여, 스펙트럼 값(s) 또는 스펙트럼 값의 최상위 비트플레인(m)을 인코딩된 형태로 표현하는 코드 값(acod_m; value)의 스펙트럼 값(a) 또는 스펙트럼 값의 최상위 비트플레인(m)을 디코딩된 형태로 표현하는 심

볼 코드(symbol)로의 맵핑을 기술하는 맵핑 룰을 선택하는 단계를 포함하며,

상기 수치적 현재 콘텍스트 값은 이전에 디코딩된 복수의 스펙트럼 값들에 의존하여 결정되며,

상기 수치적 현재 콘텍스트 값이 테이블의 엔트리에 의해 기술된 테이블 콘텍스트 값과 동일하거나 또는 상기 테이블의 엔트리들에 의해 기술된 구간 내에 놓여 있는지 여부를 결정하고, 선택된 맵핑 룰을 기술하는 맵핑 룰 인덱스 값을 유도하도록, 반복 구간 사이즈 감소를 이용하여 적어도 하나의 테이블이 평가되는, 디코딩된 오디오 정보를 제공하기 위한 방법.

청구항 13

입력 오디오 정보에 기초하여 인코딩된 오디오 정보를 제공하기 위한 방법에 있어서,

주파수 영역 오디오 표현이 스펙트럼 값들의 세트를 포함하도록, 에너지 압축 시간 영역-대-주파수 영역 변환을 이용하여 입력 오디오 정보의 시간 영역 표현에 기초하여 주파수 영역 오디오 표현을 제공하는 단계; 및

스펙트럼 값, 또는 스펙트럼 값의 사전처리된 버전을 가변 길이 코드워드를 이용하여 산술적으로 인코딩하는 단계

를 포함하며, 스펙트럼 값 또는 스펙트럼 값의 최상위 비트플레인의 값은 코드 값에 맵핑되고,

현재 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값에 의존하여 코드 값으로의 스펙트럼 값, 또는 스펙트럼 값의 최상위 비트플레인의 맵핑을 기술하는 맵핑 룰이 선택되고,

상기 수치적 현재 콘텍스트 값은 이전에 디코딩된 복수의 스펙트럼 값들에 의존하여 결정되며,

상기 수치적 현재 콘텍스트 값이 테이블의 엔트리에 의해 기술된 테이블 콘텍스트 값과 동일하거나 또는 상기 테이블의 엔트리들에 의해 기술된 구간 내에 놓여 있는지 여부를 결정하고, 선택된 맵핑 룰을 기술하는 맵핑 룰 인덱스 값을 결정하도록, 반복 구간 사이즈 감소를 이용하여 적어도 하나의 테이블이 평가되는, 인코딩된 오디오 정보를 제공하기 위한 방법.

청구항 14

컴퓨터에서 실행될 때 제12항 또는 제13항에 따른 방법을 수행하기 위한 컴퓨터 프로그램을 저장한 컴퓨터-판독가능 저장 매체.

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 말하는 통합형 음성 및 오디오 코더(unified speech and audio coder; USAC)와 같은 오디오 인코더 또는 디코더에서 이용될 수 있는 개선된 스펙트럼 무잡음 코딩에 관한 것이다.

배 경 기 술

[0003] 이하에서는, 본 발명과 본 발명의 장점들의 이해를 용이하게 하기 위해 본 발명의 배경기술을 간략하게 설명할 것이다. 과거 수 십년 동안, 오디오 콘텐츠를 디지털방식으로 저장하고 양호한 비트레이트 효율성을 가지면서 배포하는 가능성을 창출하는데에 많은 노력을 쏟아왔었다. 이러한 방식에서의 한가지 중요한 달성은 국제 표준 ISO/IEC 14496-3의 정의이다. 이 표준의 파트 3은 오디오 콘텐츠의 인코딩과 디코딩에 관한 것이며, 파트 3의 서브파트 4는 일반적인 오디오 코딩에 관한 것이다. ISO/IEC 14496 파트 3에서, 서브파트 4는 일반적인 오디오 콘텐츠의 인코딩 및 디코딩에 관한 개념을 정의한다. 추가로, 퀄리티를 향상시키고 및/또는 필수 비트레이트를 감소시키기 위해 추가적인 개선책들이 제안되어 왔다.

[0004] 상기 표준에서 기술된 개념에 따르면, 시간 영역 오디오 신호는 시간 주파수 표현으로 전환된다. 시간 영역으로부터 시간 주파수 영역으로의 변환은 일반적으로 변환 블록들을 이용하여 수행되는데, 이 변환 블록은 시간 영

역 샘플들의 "프레임"으로서 칭해지기도 한다. 프레임의 절반 만큼 쉬프트되어 오버랩된 프레임들을 이용하는 것은 유리하다는 것이 발견되어 왔는데, 그 이유는 오버랩은 인공물(artifact)을 효과적으로 방지(또는 적어도 감소)시키기 때문이다. 또한, 일시적으로 제한된 프레임들의 이러한 처리로부터 발생하는 인공물들을 방지하기 위해서는 윈도우잉(windowing)이 수행되어야 한다는 것이 발견되어 왔다.

[0005] 입력 오디오 신호의 윈도우잉된 부분을 시간 영역으로부터 시간 주파수 영역으로 변환시킴으로써 많은 경우들에서 에너지 압축(energy compaction)이 획득되었으며 이로써 몇몇의 스펙트럼 값들은 복수의 다른 스펙트럼 값들보다 상당히 큰 크기를 갖는다. 따라서, 많은 경우들에서, 스펙트럼 값들의 평균 크기보다 상당히 큰 크기를 갖는 상대적으로 작은 수의 스펙트럼 값들이 존재한다. 에너지 압축을 불러일으키는 시간 영역으로부터 시간 주파수 영역으로의 변환의 일반적인 예시는 소위 말하는 변형 이산 코사인 변환(modified discrete cosine transform; MDCT)이다.

[0006] 심리음향적으로 보다 중요한 스펙트럼 값들에 대해서는 양자화 에러가 상대적으로 작도록 하고, 심리음향적으로 덜 중요한 스펙트럼 값들에 대해서는 양자화 에러가 상대적으로 크도록, 스펙트럼 값들은 종종 심리음향적(psychoacoustic) 모델에 따라 스케일링되고 양자화된다. 스케일링되고 양자화된 스펙트럼 값들은 자신들의 비트레이트 효율적인 표현을 제공하기 위해 인코딩된다.

[0007] 예를 들어, 양자화된 스펙트럼 계수들의 소위 말하는 호프만 코딩의 이용이 국제 표준 ISO/IEC 14496-3:2005(E), 파트 3, 서브파트 4에서 기술된다.

[0008] 하지만, 스펙트럼 값들의 코딩의 퀄리티는 필수 비트레이트에 상당한 영향을 미친다는 것이 발견되어 왔다. 또한, 휴대형 가전 제품에서 종종 구현됨에 따라 값싸고 저전력 소모형이어야 하는 오디오 디코더의 복잡성은 스펙트럼 값들을 인코딩하는데 이용되는 코딩에 의존적이라는 것이 발견되어 왔다.

발명의 내용

해결하려는 과제

[0009] 이러한 상황을 비추어 보면, 비트레이트 효율성과 계산적인 수고로움간의 개선된 트레이드오프를 제공해주는 오디오 콘텐츠의 인코딩 및 디코딩에 대한 개념이 필요하다.

과제의 해결 수단

[0010] 본 발명에 따른 실시예는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더를 생성한다. 오디오 디코더는 스펙트럼 계수들의 산술적으로 인코딩된 표현에 기초하여 복수의 디코딩된 스펙트럼 값들을 제공하는 산술 디코더를 포함한다. 산술 디코더는 또한 디코딩된 오디오 정보를 획득하기 위해, 디코딩된 스펙트럼 값들을 이용하여 시간 영역 오디오 표현을 제공하는 주파수 영역-대-시간 영역(frequency-domain-to-time-domain) 컨버터를 포함한다. 산술 디코더는 현재 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값(numeric current context value)에 의존하여 심볼 코드로의 코드 값의 맵핑을 기술하는 맵핑 룰을 선택하도록 구성된다. 산술 디코더는 이전에 디코딩된 복수의 스펙트럼 값들에 의존하여 수치적 현재 콘텍스트 값을 결정하도록 구성된다. 또한, 산술 디코더는 반복 구간 사이즈 감소를 이용하여 적어도 하나의 테이블을 평가하고, 선택된 맵핑 룰을 기술하는 맵핑 룰 인덱스 값을 유도하기 위해, 수치적 현재 콘텍스트 값이 테이블의 엔트리에 의해 기술된 테이블 콘텍스트 값과 동일하거나 또는 테이블의 엔트리들에 의해 기술된 구간 내에 놓여 있는지 여부를 결정하도록 구성된다.

[0011] 본 발명에 따른 실시예는 오디오 콘텐츠의 스펙트럼 값들을 디코딩하기 위한 산술 디코더의 현재 콘텍스트 상태를 기술하고, 맵핑 룰 인덱스 값의 유도에 적합한 수치적 현재 콘텍스트 값을 제공하는 것이 가능하다는 것을 발견한 것에 기초하며, 여기서의 맵핑 룰 인덱스 값은 테이블에 기초한 반복 구간 사이즈 감소를 이용하여 산술 디코더에서 선택될 맵핑 룰을 기술한다. 반복 구간 사이즈 감소를 이용한 테이블 검색은 일반적으로 비교적 많은 수의 상이한 콘텍스트 상태들을 기술하도록 계산된, 수치적 현재 콘텍스트 값에 의존하여, 비교적 작은 수의 맵핑 룰들 중에서 (맵핑 룰 인덱스 값에 의해 기술된) 맵핑 룰을 선택하는데 적합하다는 것이 발견되어 왔으며, 잠재적 맵핑 룰의 갯수는 일반적으로 수치적 현재 콘텍스트 값에 의해 기술된 잠재적 콘텍스트 상태들의 갯수보다 적어도 10배만큼 작다. 상세한 분석은 반복 구간 사이즈 감소를 이용함으로써 적절한 맵핑 룰의 선택이 높은 계산 효율성을 갖고 수행될 수 있다는 것을 보여줬다. 테이블 액세스의 횟수는 최악의 경우에서조차도, 이러한 개념에 의해 비교적 작게 유지될 수 있다. 이것은 실시간 환경에서 오디오 디코딩을 구현하려고 시도할 때에 매우 긍정적인 것으로 나타났다. 뿐만 아니라, 반복 구간 사이즈 감소는 수치적 현재 콘텍스트 값이 테이블의 엔

트리에 의해 기술된 테이블 콘텍스트 값과 동일한지 여부의 검출과 테이블의 엔트리들에 의해 기술된 구간 내에 수치적 현재 콘텍스트 값이 놓여 있는지 여부의 검출 모두에 적용될 수 있다는 것이 발견되어 왔다.

[0012] 요약하자면, 반복 구간 사이즈 감소의 이용은 수치적 현재 콘텍스트 값에 의존하여 오디오 콘텐츠의 산술 디코딩을 위한 맵핑 룰을 선택하기 위해 해싱 알고리즘을 수행하는데 적합하다는 것이 발견되어 왔는데, 일반적으로 맵핑 룰들의 저장소에 대한 메모리 요건을 상당히 작게 유지하기 위해 수치적 현재 콘텍스트 값의 잠재적 값들의 수는 맵핑 룰의 수보다 상당히 크다.

[0013] 바람직한 실시예에서, 산술 디코더는 초기 테이블 구간의 하위 경계를 지정하기 위해 하위 구간 경계 변수를 초기화하고 초기 테이블 구간의 상위 경계를 지정하기 위해 상위 구간 경계 변수를 초기화하도록 구성된다. 또한 산술 디코더는 바람직하게 초기 테이블 구간의 중심에 테이블 인덱스가 배열되어 있는 테이블 엔트리를 평가하고 이 평가된 테이블 엔트리에 의해 표현된 테이블 콘텍스트 값과 수치적 현재 콘텍스트 값을 비교하도록 구성된다. 산술 디코더는 또한 업데이트된 테이블 구간을 획득하기 위해, 이러한 비교의 결과에 의존하여 하위 구간 경계 변수 또는 상위 구간 경계 변수를 조정하도록 구성된다. 또한, 산술 디코더는, 테이블 콘텍스트 값이 수치적 현재 콘텍스트 값과 동일하거나 또는 업데이트된 구간 경계 변수들에 의해 정의된 테이블 구간의 사이즈가 테이블 구간 사이즈 문턱값(threshold)에 도달하거나 또는 그 아래로 내려갈 때 까지, 하나 이상의 업데이트된 테이블 구간들에 기초하여 테이블 엔트리의 평가 및 하위 구간 경계 변수 또는 상위 구간 경계 변수의 조정을 반복하도록 구성된다. 반복 구간 사이즈 감소는 상술한 단계들을 이용하여 효율적으로 구현될 수 있다는 것이 발견되었다.

[0014] 바람직한 실시예에서, 산술 디코더는 테이블의 주어진 엔트리가 수치적 현재 콘텍스트 값과 동일한 테이블 콘텍스트 값을 나타낸다는 발견에 응답하여 테이블의 상기 주어진 엔트리에 의해 기술된 맵핑 룰 인덱스 값을 제공하도록 구성된다. 이에 따라, 일반적으로 시간과 전기 에너지를 소모시키는 테이블 액세스들의 횟수가 작게 유지되기 때문에, 하드웨어 구현에 적합한 매우 효율적인 테이블 액세스 메커니즘이 구현된다.

[0015] 바람직한 실시예에서, 산술 디코더는 준비 단계들에서 하위 구간 경계 변수 i_min 가 -1로 설정되고 상위 구간 경계 변수 i_max 가 테이블 엔트리들의 갯수에 1을 뺀 수로 설정되는 알고리즘을 수행하도록 구성된다. 알고리즘에서, 구간 경계 변수 i_max 와 구간 경계 변수 i_min 간의 차이가 1보다 큰지 여부를 추가로 체크하고, 후속 단계들은 상기 언급한 조건 (i_max - i_min>1) 이 더 이상 충족되지 않거나 또는 중지 조건에 도달될 때 까지, 다음의 단계들, 즉 (1) 변수 i 를 i_min + ((i_max - i_min)/2)로 설정하는 단계, (2) 테이블 인덱스 i 를 갖는 테이블 엔트리에 의해 기술된 테이블 콘텍스트 값이 수치적 현재 콘텍스트 값보다 큰 경우 상위 구간 경계 변수 i_max 를 i 로 설정하는 단계, 및 (3) 테이블 인덱스 i 를 갖는 테이블 엔트리에 의해 기술된 테이블 콘텍스트 값이 수치적 현재 콘텍스트 값보다 작은 경우 하위 구간 경계 변수 i_min 를 i 로 설정하는 단계를 반복한다. 앞서 설명한 단계들 (1), (2), (3)의 반복은, 테이블 인덱스 i 를 갖는 테이블 엔트리에 의해 기술된 테이블 콘텍스트 값이 수치적 현재 콘텍스트 값과 동일한 경우에 중지된다. 이 경우, 즉 테이블 인덱스 i 를 갖는 테이블 엔트리에 의해 기술된 테이블 콘텍스트 값이 수치적 현재 콘텍스트 값과 동일한 경우, 테이블 인덱스 i 를 갖는 테이블 엔트리에 의해 기술된 맵핑 룰 인덱스 값은 반환된다. 오디오 디코더에서의 이러한 알고리즘의 실행은 맵핑 룰을 선택할 때에 매우 우수한 계산 효율성을 제공해준다.

[0016] 바람직한 실시예에서, 산술 디코더는 이전에 디코딩된 스펙트럼 값들의 크기들을 기술하는 크기 값들의 가중화된 조합에 기초하여 수치적 현재 콘텍스트 값을 획득하도록 구성된다. 수치적 현재 콘텍스트 값을 획득하기 위한 이러한 메커니즘은 반복 구간 사이즈 감소를 이용하여 맵핑 룰의 효율적인 선택을 가능하게 해주는 수치적 현재 콘텍스트 값을 초래시킨다는 것이 발견되었다. 이것은, 수치적으로 인접해 있는 수치적 현재 콘텍스트 값들이 종종 현재 디코딩되는 스펙트럼 값의 유사한 콘텍스트 환경들에 관련되도록, 이전에 디코딩된 스펙트럼 값들의 크기들을 기술하는 크기 값들의 가중화된 조합은 수치적 현재 콘텍스트 값을 초래시킨다는 사실에 기인한다. 이것은 반복 구간 사이즈 감소에 기초한 해싱 알고리즘의 효율적인 적용을 가능하게 해준다.

[0017] 바람직한 실시예에서, 테이블은 복수의 엔트리들을 포함하며, 복수의 엔트리들 각각은 테이블 콘텍스트 값 및 연관된 맵핑 룰 인덱스 값을 기술하며, 테이블의 엔트리들은 테이블 콘텍스트 값들에 따라 수치적으로 순서화된다. 이러한 테이블은 반복 구간 사이즈 감소와 결합된 응용에 매우 적합하다는 것이 발견되었다. 테이블의 엔트리들의 수치적 순서화는 수치적 현재 콘텍스트 값이 놓여 있는 구간의 확인과 함께, 수치적 현재 콘텍스트 값과 동일한 테이블 콘텍스트 값의 검색을 비교적 작은 반복 횟수 내에서 수행하도록 해준다. 따라서, 테이블 액세스들의 횟수는 작게 유지된다. 또한, 테이블 콘텍스트 값과 단일 테이블 엔트리 내의 연관된 맵핑 룰 인덱스 값을 결합함으로써, 테이블 액세스의 횟수는 감소될 수 있는데, 이것은 하드웨어 장치에서의 실행 시간과 장치의 전

력 소모를 작게 유지시키는데 도움을 준다.

[0018] 바람직한 실시예에서, 테이블은 복수의 엔트리들을 포함하며, 복수의 엔트리들 각각은 콘텍스트 값 구간의 경계 값을 정의하는 테이블 콘텍스트 값과, 콘텍스트 값 구간과 연관된 맵핑 룰 인덱스 값을 기술한다. 이 개념을 이용하여, 수치적 현재 콘텍스트 값이 놓여 있는 구간을 반복 구간 사이즈 감소를 이용하여 효율적으로 확인하는 것이 가능하다. 다시, 반복 횟수와 테이블 액세스 횟수는 작게 유지될 수 있다.

[0019] 바람직한 실시예에서, 산술 디코더는 수치적 현재 콘텍스트 값에 의존하여 두 개의 맵핑 룰 선택 단계를 수행하도록 구성된다. 이 경우, 산술 디코더는, 제1 선택 단계에서, 수치적 현재 콘텍스트 값, 또는 이로부터 유도된 값이 다이렉트 히트(direct-hit) 테이블의 엔트리에 의해 기술된 중요 상태 값(significant state value)과 동일한지 여부를 체크하도록 구성된다. 산술 디코더는 또한, 수치적 현재 콘텍스트 값, 또는 이로부터 유도된 값이 다이렉트 히트 테이블의 엔트리들에 의해 기술된 중요 상태 값들과 상이한 경우에만 실행되는 제2 선택 단계에서, 복수의 구간들 중에서 어느 구간에 수치적 현재 콘텍스트 값이 놓여 있는지를 결정하도록 구성된다. 산술 디코더는 반복 구간 사이즈 감소를 이용하여 다이렉트 히트 테이블을 평가하고, 수치적 현재 콘텍스트 값이 다이렉트 히트 테이블의 엔트리에 의해 기술된 테이블 콘텍스트 값과 동일한지 여부를 결정하도록 구성된다. 이러한 두 단계의 테이블 평가 메커니즘을 이용함으로써 특별히 중요한 콘텍스트 상태들(이 특별히 중요한 콘텍스트 상태들은 다이렉트 히트 테이블의 엔트리들에 의해 기술됨)을 효율적으로 확인하고, 또한 제2 선택 단계에서 덜 중요한 콘텍스트 상태들(이 상태들은 다이렉트 히트 테이블의 엔트리들에 의해 기술되지 않음)을 위한 적절한 맵핑 룰을 선택하는 것이 가능하다는 것을 발견하였다. 이렇게 함으로써, 가장 중요한 콘텍스트 상태들은 제1 선택 단계에서 처리될 수 있는데, 이것은 특별히 중요한 상태의 존재시에 계산적 복잡도를 감소시킨다. 더군다나, 덜 중요한 상태들에 대해서도 적합한 맵핑 룰을 찾는 것이 가능하다.

[0020] 바람직한 실시예에서, 산술 디코더는, 제2 선택 단계에서, 구간 맵핑 테이블을 평가하도록 구성되며, 이 테이블의 엔트리들은 반복 구간 사이즈 감소를 이용하여 콘텍스트 값 구간들의 경계 값들을 기술한다. 반복 구간 사이즈 감소는 다이렉트 히트의 확인 및 구간 맵핑 테이블에 의해 기술된 복수의 구간들 중에서 수치적 현재 콘텍스트 값이 놓여 있는 구간의 확인 모두에 대해서 적합하다는 것이 발견되었다.

[0021] 바람직한 실시예에서, 산술 디코더는, 테이블 구간의 사이즈가 미리결정된 테이블 구간 사이즈 문턱값에 도달하거나 그 아래로 감소하거나 또는 테이블 구간의 중심에서 테이블 엔트리에 의해 기술된 구간 경계 콘텍스트 값이 수치적 현재 콘텍스트 값과 동일할 때 까지, 구간 맵핑 테이블의 엔트리들에 의해 표현된 구간 경계 콘텍스트 값들과 수치적 현재 콘텍스트 값간의 비교에 의존하여 테이블 구간의 사이즈를 반복적으로 감소시키도록 구성된다. 산술 디코더는 테이블 구간의 반복적인 감소가 회피될 때 테이블 구간의 구간 경계의 설정에 의존하여 맵핑 룰 인덱스 값을 제공하도록 구성된다. 이러한 개념을 이용함으로써, 구간 맵핑 테이블의 엔트리들에 의해 정의된 복수의 테이블 구간들 중 수치적 현재 콘텍스트 값이 놓여 있는 테이블 구간을 낮은 계산적 수고로움을 갖고 결정할 수 있다. 따라서, 맵핑 룰은 낮은 계산적 수고로움을 갖고 선택될 수 있다.

[0022] 본 발명에 따른 실시예는 입력 오디오 정보에 기초하여 인코딩된 오디오 정보를 제공하기 위한 오디오 인코더를 생성한다. 오디오 인코더는 주파수 영역 오디오 표현이 스펙트럼 값들의 세트를 포함하도록, 입력 오디오 정보의 시간 영역 표현에 기초하여 주파수 영역 오디오 표현을 제공하기 위한 에너지 압축 시간 영역-대-주파수 영역 컨버터를 포함한다. 오디오 인코더는 또한 가변 길이 코드워드를 이용하여 스펙트럼 값 또는 이것의 사전처리된 버전을 인코딩하도록 구성된 산술 인코더를 포함한다. 산술 인코더는 스펙트럼 값, 또는 스펙트럼 값의 최상위 비트플레인의 값을 코드 값에 맵핑하도록 구성된다. 산술 인코더는 현재 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값에 의존하여 코드 값으로의 스펙트럼 값, 또는 스펙트럼 값의 최상위 비트플레인의 맵핑을 기술하는 맵핑 룰을 선택하도록 구성된다. 산술 인코더는 이전에 인코딩된 복수의 스펙트럼 값들에 의존하여 수치적 현재 콘텍스트 값을 결정하도록 구성된다. 산술 인코더는 반복 구간 사이즈 감소를 이용하여 적어도 하나의 테이블을 평가하고, 수치적 현재 콘텍스트 값이 테이블의 엔트리에 의해 기술된 콘텍스트 값과 동일하거나 또는 테이블의 엔트리들에 의해 기술된 구간 내에 놓여 있는지 여부를 결정하고, 이로써 선택된 맵핑 룰을 기술하는 맵핑 룰 인덱스 값을 유도하도록 구성된다. 이 오디오 신호 인코더는 상술한 오디오 신호 디코더와 동일한 발견에 기초한다. 오디오 콘텐츠의 디코딩에 효율적인 것으로 나타난 맵핑 룰의 선택을 위한 메커니즘은 또한, 일관된 시스템을 허용하기 위해, 인코더측에서도 적용되어야 한다는 것이 발견되었다.

[0023] 본 발명에 따른 실시예는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 방법을 생성한다.

[0024] 본 발명에 따른 또 다른 실시예는 입력 오디오 정보에 기초하여 인코딩된 오디오 정보를 제공하기 위한 방법을

생성한다.

[0025] 본 발명에 따른 또 다른 실시예는 상기 방법들 중 하나의 방법을 수행하기 위한 컴퓨터 프로그램을 생성한다.

[0026] 본 방법 및 컴퓨터 프로그램은 상술한 오디오 디코더 및 상술한 오디오 인코더와 동일한 발견들에 기초한다.

도면의 간단한 설명

[0027] 이하에서는 첨부된 도면들을 참조하면서 본 발명에 따른 실시예들을 설명한다:

도 1은 본 발명의 실시예에 따른, 오디오 인코더의 개략적인 블록도를 도시한다.

도 2는 본 발명의 실시예에 따른, 오디오 디코더의 개략적인 블록도를 도시한다.

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

도 4는 상태 계산을 위한 콘텍스트의 개략적인 표현을 도시한다.

도 5a는 콘텍스트를 맵핑하기 위한 알고리즘 "arith_map_context ()"의 의사 프로그램 코드 표현을 도시한다.

도 5b와 도 5c는 콘텍스트 상태 값을 획득하기 위한 알고리즘 "arith_get_context ()"의 의사 프로그램 코드 표현을 도시한다.

도 5d는 상태 변수로부터 누적 도수 테이블(cumulative-frequency table) 인덱스 값 "pki"을 유도하기 위한 알고리즘 "get_pk(s)"의 의사 프로그램 코드 표현을 도시한다.

도 5e는 상태 값으로부터 누적 도수 테이블 인덱스 값 "pki"을 유도하기 위한 알고리즘 "arith_get_pk(s)"의 의사 프로그램 코드 표현을 도시한다.

도 5f는 상태 값으로부터 누적 도수 테이블 인덱스 값 "pki"을 유도하기 위한 알고리즘 "get_pk(unsigned long s)"의 의사 프로그램 코드 표현을 도시한다.

도 5g는 가변 길이 코드워드로부터 심볼을 산술적으로 디코딩하기 위한 알고리즘 "arith_decode ()"의 의사 프로그램 코드 표현을 도시한다.

도 5h는 콘텍스트를 업데이트하기 위한 알고리즘 "arith_update_context ()"의 의사 프로그램 코드 표현을 도시한다.

도 5i는 정의들 및 변수들의 범례를 도시한다.

도 6a는 통합형 음성 및 오디오 코딩(nified speech and audio coding; USAC) 미가공 데이터 블록의 구문(syntax) 표현을 도시한다.

도 6b는 단일 채널 엘리먼트의 구문 표현을 도시한다.

도 6c는 채널 쌍 엘리먼트의 구문 표현을 도시한다.

도 6d는 "ics" 제어 정보의 구문 표현을 도시한다.

도 6e는 주파수 영역 채널 스트림의 구문 표현을 도시한다.

도 6f는 산술적으로 코딩된 스펙트럼 데이터의 구문 표현을 도시한다.

도 6g는 스펙트럼 값들의 세트를 디코딩하기 위한 구문 표현을 도시한다.

도 6h는 데이터 엘리먼트들 및 변수들의 범례를 도시한다.

도 7은 본 발명의 또 다른 실시예에 따른, 오디오 인코더의 개략적인 블록도를 도시한다.

도 8은 본 발명의 또 다른 실시예에 따른, 오디오 디코더의 개략적인 블록도를 도시한다.

도 9는 본 발명에 따른 코딩 방식과 USAC 드래프트 표준의 작업 드래프트 3에 따른 무잡음 코딩의 비교를 위한 장치를 도시한다.

도 10a는 USAC 드래프트 표준의 작업 드래프트 4에 따라 이용될 때의, 상태 계산을 위한 콘텍스트의 개략도를

도시한다.

도 10b는 본 발명에 따른 실시예들에서 이용될 때의, 상태 계산을 위한 콘텍스트의 개략도를 도시한다.

도 11a는 USAC 드래프트 표준의 작업 드래프트 4에 따른 산술 코딩 방식에서 이용되는 테이블의 개관을 도시한다.

도 11b는 본 발명에 따른 산술 코딩 방식에서 이용되는 테이블의 개관을 도시한다.

도 12a는 USAC 드래프트 표준의 작업 드래프트 4 및 본 발명에 따른 무잡음 코딩 방식들에 대한 판독 전용 메모리(ROM) 수요량의 그래픽 표현을 도시한다.

도 12b는 USAC 드래프트 표준의 작업 드래프트 4에 따른 개념과 본 발명에 따른 총 USAC 디코더 데이터 판독 전용 메모리(ROM) 수요량의 그래픽 표현을 도시한다.

도 13a는 본 발명의 실시예에 따른 산술 디코더 및 USAC 드래프트 표준의 작업 드래프트 3에 따른 산술 코더를 이용한, 통합형 음성 및 오디오 코딩 코더에 의해 이용되는 평균 비트레이트들의 테이블 표현을 도시한다.

도 13b는 본 발명의 실시예에 따른 산술 코더 및 USAC 드래프트 표준의 작업 드래프트 3에 따른 산술 코더를 이용한, 통합형 음성 및 오디오 코딩 코더를 위한 비트저장소 제어의 테이블 표현을 도시한다.

도 14는 본 발명의 실시예, 및 USAC 드래프트 표준의 작업 드래프트 3에 따른 USAC 코더를 위한 평균 비트레이트들의 테이블 표현을 도시한다.

도 15는 USAC의 프레임 단위의 최소, 최대 및 평균 비트레이트들의 테이블 표현을 도시한다.

도 16은 프레임 단위의 최상의 경우 및 최악의 경우의 테이블 표현을 도시한다.

도 17a 및 도 17b는 테이블 “ari_s_hash[387]”의 콘텐츠의 테이블 표현을 도시한다.

도 18은 테이블 “ari_gs_hash[225]”의 콘텐츠의 테이블 표현을 도시한다.

도 19a 및 도 19b는 테이블 “ari_cf_m[64][9]”의 콘텐츠의 테이블 표현을 도시한다.

도 20a 및 도 20b는 테이블 "ari_s_hash[387]"의 콘텐츠의 테이블 표현을 도시한다.

도 21은 본 발명의 실시예에 따른, 오디오 인코더의 개략적인 블록도를 도시한다.

도 22는 본 발명의 실시예에 따른, 오디오 디코더의 개략적인 블록도를 도시한다.

발명을 실시하기 위한 구체적인 내용

[0028] 1. 도 7에 따른 오디오 인코더

[0029] 도 7은 본 발명의 실시예에 따른, 오디오 인코더의 개략적인 블록도를 도시한다. 오디오 인코더(700)는 입력 오디오 정보(710)를 수신하고, 이를 기초로, 인코딩된 오디오 정보(712)를 제공하도록 구성된다. 오디오 인코더는 주파수 영역 오디오 표현(722)이 스펙트럼 값들의 세트를 포함하도록, 입력 오디오 정보(710)의 시간 영역 표현에 기초하여 주파수 영역 오디오 표현(722)을 제공하도록 구성된 에너지 압축 시간 영역-대-주파수 영역 컨버터(720)를 포함한다. 오디오 인코더(700)는 또한 (주파수 영역 오디오 표현(722)을 형성하는 스펙트럼 값들의 세트 중에서) 스펙트럼 값, 또는 이 스펙트럼 값의 사전처리된 버전을 가변 길이 코드워드를 이용하여 인코딩하여 인코딩된 오디오 정보(이것은 예컨대 복수의 가변 길이 코드워드를 포함할 수 있음)(712)를 획득하도록 구성된 산술 인코더(730)를 포함한다.

[0030] 산술 인코더(730)는 콘텍스트 상태에 의존하여, 스펙트럼 값, 또는 스펙트럼 값의 최상위 비트플레인(most-significant bit-plane)의 값을 코드 값에 맵핑(즉, 가변 길이 코드워드에 맵핑)하도록 구성된다. 산술 인코더(730)는 콘텍스트 상태에 의존하여, 코드 값으로의 스펙트럼 값 또는 스펙트럼 값의 최상위 비트플레인의 맵핑을 기술하는 맵핑 룰을 선택하도록 구성된다. 산술 인코더는 이전에 인코딩된 (필수적이지는 않지만 바람직하게는, 인접해 있는) 복수의 스펙트럼 값들에 의존하여 현재 콘텍스트 상태를 결정하도록 구성된다. 이를 목적으로, 산술 인코더는 이전에 인코딩된 인접해 있는 복수의 스펙트럼 값들의 그룹(이 스펙트럼 값들의 그룹은 개별적으로 또는 다함께, 각자의 크기에 관한 미리결정된 조건을 충족시킴)을 검출하고, 이러한 검출의 결과에 의존하여 현재 콘텍스트 상태를 결정하도록 구성된다.

[0031] 살펴볼 수 있는 바와 같이, 코드 값으로의 스펙트럼 값 또는 스펙트럼 값의 최상위 비트플레인의 맵핑은 맵핑

룰(742)을 이용하여 스펙트럼 값 인코딩(740)에 의해 수행될 수 있다. 상태 추적기(750)는 콘텍스트 상태를 추적하도록 구성될 수 있고, 각자의 크기에 관한 미리결정된 조건을 개별적으로 또는 다함께 충족시키는 이전에 인코딩된 인접한 복수의 스펙트럼 값들의 그룹을 검출하기 위한 그룹 검출기(752)를 포함할 수 있다. 상태 추적기(750)는 또한 바람직하게는 그룹 검출기(752)에 의해 수행된 상기 검출의 결과에 의존하여 현재 콘텍스트 상태를 결정하도록 구성된다. 따라서, 상태 추적기(750)는 현재 콘텍스트 상태를 기술하는 정보(754)를 제공한다. 맵핑 룰 선택기(760)는 코드 값으로의 스펙트럼 값의 맵핑 또는 코드 값으로의 스펙트럼 값의 최상위 비트 플레인의 맵핑을 기술하는 맵핑 룰, 예컨대 누적 도수 테이블을 선택할 수 있다. 따라서, 맵핑 룰 선택기(760)는 맵핑 룰 정보(742)를 스펙트럼 인코딩(740)에게 제공한다.

[0032] 위를 요약하자면, 오디오 인코더(700)는 시간 영역-대-주파수 영역 컨버터에 의해 제공된 주파수 영역 오디오 표현의 산술 인코딩을 수행한다. 산술 인코딩은, 맵핑 룰(예컨대, 누적 도수 테이블(cumulative-frequencies-table))이 이전에 인코딩된 스펙트럼 값들에 의존하여 선택되도록 콘텍스트 의존적이다. 따라서, 시간적으로 및/또는 주파수적으로 (또는, 적어도 미리결정된 환경 내에서) 서로 인접해 있고 및/또는 현재 인코딩된 스펙트럼 값(즉, 현재 인코딩된 스펙트럼 값의 미리결정된 환경 내의 스펙트럼 값들)과 인접해 있는 스펙트럼 값들은 산술 인코딩에 의해 평가된 확률 분포를 조정하도록 산술 인코딩에서 고려된다. 적절한 맵핑 룰을 선택할 때, 각자의 크기에 관한 미리결정된 조건을 개별적으로 또는 다함께 충족시키는 이전에 인코딩된 인접한 복수의 스펙트럼 값들의 그룹이 존재하는지 여부를 검출하기 위한 검출이 수행된다. 이러한 검출의 결과는 현재 콘텍스트 상태의 선택, 즉 맵핑 룰의 선택에서 적용된다. 특별히 작거나 특별히 큰 복수의 스펙트럼 값들의 그룹이 존재하는지 여부를 검출함으로써, 시간 주파수 표현일 수 있는, 주파수 영역 오디오 표현 내에서 특정한 특징들을 인식하는 것이 가능하다. 예컨대 특별히 작거나 특별히 큰 복수의 스펙트럼 값들의 그룹과 같은 특정한 특징들은, 특정한 콘텍스트 상태가 특별히 우수한 코딩 효율성을 제공할 수 있으므로 이러한 특정한 콘텍스트 상태가 이용되어야 한다는 것을 표시한다. 따라서, 이전에 코딩된 복수의 스펙트럼 값들의 조합에 기초한 대안적인 콘텍스트 평가와 결합하여 일반적으로 이용되는, 미리결정된 조건을 충족시키는 인접한 스펙트럼 값들의 그룹의 검출은 입력 오디오 정보가 몇몇의 특정 상태들(예컨대 크게 마스킹된 주파수 범위를 포함함)을 취한 경우에 적절한 콘텍스트의 효율적인 선택을 가능하게 해주는 메커니즘을 제공한다.

[0033] 따라서, 콘텍스트 계산을 충분히 단순하게 유지시키면서 효율적인 인코딩이 달성될 수 있다.

[0034] 2. 도 8에 따른 오디오 디코더

[0035] 도 8은 오디오 디코더(800)의 개략적인 블록도를 도시한다. 오디오 디코더(800)는 인코딩된 오디오 정보(810)를 수신하고, 이를 기초로, 디코딩된 오디오 정보(812)를 제공하도록 구성된다. 오디오 디코더(800)는 스펙트럼 값들의 산술적으로 인코딩된 표현(821)에 기초하여 복수의 디코딩된 스펙트럼 값들(822)을 제공하도록 구성된 산술 디코더(820)를 포함한다. 오디오 디코더(800)는 또한 디코딩된 스펙트럼 값들(822)을 수신하고, 디코딩된 오디오 정보(812)를 획득하기 위해, 디코딩된 스펙트럼 값들(822)을 이용하여, 디코딩된 오디오 정보를 구성할 수 있는 시간 영역 오디오 표현(812)을 제공하도록 구성된 주파수 영역-대-시간 영역 컨버터(830)를 포함한다.

[0036] 산술 디코더(820)는 스펙트럼 값들의 산술적으로 인코딩된 표현(821)의 코드 값을 하나 이상의 디코딩된 스펙트럼 값들, 또는 하나 이상의 디코딩된 스펙트럼 값들의 적어도 일부분(예컨대, 최상위 비트 플레인)을 표현하는 심볼 코드로 맵핑하도록 구성된 스펙트럼 값 결정기(824)를 포함한다. 스펙트럼 값 결정기(824)는 맵핑 룰 정보(828a)에 의해 기술될 수 있는 맵핑 룰에 의존하여 맵핑을 수행하도록 구성될 수 있다.

[0037] 산술 디코더(820)는 (콘텍스트 상태 정보(826a)에 의해 기술될 수 있는) 콘텍스트 상태에 의존하여 (하나 이상의 스펙트럼 값들을 기술하는) 심볼 코드로의 (스펙트럼 값들의 산술적으로 인코딩된 표현(821)에 의해 기술된) 코드 값의 맵핑을 기술하는 맵핑 룰(예컨대, 누적 도수 테이블)을 선택하도록 구성된다. 산술 디코더(820)는 이전에 디코딩된 복수의 스펙트럼 값들(822)에 의존하여 현재 콘텍스트 상태를 결정하도록 구성된다. 이를 위해, 이전에 디코딩된 스펙트럼 값들을 기술하는 정보를 수신하는 상태 추적기(826)가 이용될 수 있다. 산술 디코더는 또한 각자의 크기에 관한 미리결정된 조건을 개별적으로 또는 다함께 충족시키는 이전에 디코딩된 (필수적이지는 않지만 바람직하게는, 인접해 있는) 복수의 스펙트럼 값들의 그룹을 검출하고, 이러한 검출의 결과에 의존하여 (예컨대, 콘텍스트 상태 정보(826a)에 의해 기술된) 현재 콘텍스트 상태를 결정하도록 구성된다.

[0038] 각자의 크기에 관한 미리결정된 조건을 충족시키는 이전에 디코딩된 복수의 인접한 스펙트럼 값들의 그룹의 검출은, 예컨대, 상태 추적기(826)의 일부인 그룹 검출기에 의해 수행될 수 있다. 따라서, 현재 콘텍스트 상태 정보(826a)가 획득된다. 맵핑 룰의 선택은, 현재 콘텍스트 상태 정보(826a)로부터 맵핑 룰 정보(828a)를 유도하고, 맵핑 룰 정보(828a)를 스펙트럼 값 결정기(824)에게 제공하는 맵핑 룰 선택기(828)에 의해 수행될 수

있다.

[0039] 오디오 신호 디코더(800)의 기능과 관련하여, 맵핑 룰이 현재 콘텍스트 상태에 의존하여 선택되고, 이어서 현재 콘텍스트 상태는 이전에 디코딩된 복수의 스펙트럼 값들에 의존하여 결정되므로, 산술 디코더(820)는, 평균적으로, 디코딩될 스펙트럼 값에 적합한 맵핑 룰(예컨대, 누적 도수 테이블)을 선택하도록 구성된다라는 것을 유념해야 한다. 따라서, 디코딩될 인접한 스펙트럼 값들간의 통계적 의존성들이 활용될 수 있다. 더군다나, 각자의 크기에 관한 미리결정된 조건을 개별적으로 또는 다함께 충족시키는 이전에 디코딩된 인접한 복수의 스펙트럼 값들의 그룹을 검출함으로써, 맵핑 룰을 이전에 디코딩된 스펙트럼 값들의 스펙트럼 조건들(또는 패턴들)을 조정하는 것이 가능하다. 예를 들어, 이전에 디코딩된 상대적으로 작은 복수의 인접한 스펙트럼 값들의 그룹이 확인되거나, 또는 이전에 디코딩된 상대적으로 큰 복수의 인접한 스펙트럼 값들의 그룹이 확인된 경우 특정한 맵핑 룰이 선택될 수 있다. 상대적으로 큰 스펙트럼 값들의 그룹 또는 상대적으로 작은 스펙트럼 값들의 그룹의 존재는 이러한 조건에 특수하게 조정된 전용 맵핑 룰이 이용되어야 한다는 중요 표시(significant indication)로서 고려될 수 있다는 것이 발견되었다. 따라서, 콘텍스트 계산은 이러한 복수의 스펙트럼 값들의 그룹의 검출을 활용함으로써 촉진(또는 가속화)될 수 있다. 또한, 앞서 언급한 개념을 적용하지 않고서는 손쉽게 고려될 수 없는 오디오 콘텐츠의 특성들이 고려될 수 있다. 예를 들어, 각자의 크기에 관한 미리결정된 조건을 개별적으로 또는 다함께 충족시키는 복수의 스펙트럼 값들의 그룹의 검출은, 보통의 콘텍스트 계산을 위해 이용된 스펙트럼 값들의 세트와 비교할 때, 스펙트럼 값들의 상이한 세트에 기초하여 수행될 수 있다.

[0040] 자세한 내용은 아래에서 설명할 것이다.

[0041] 3. 도 1에 따른 오디오 인코더

[0042] 이하에서는, 본 발명의 실시예에 따른 오디오 인코더를 설명할 것이다. 도 1은 이러한 오디오 인코더(100)의 개략적인 블록도를 도시한다.

[0043] 오디오 인코더(100)는 입력 오디오 정보(110)를 수신하고, 이를 기초로, 인코딩된 오디오 정보를 구성하는 비트스트림(112)을 제공하도록 구성된다. 오디오 인코더(100)는 입력 오디오 정보(110)를 수신하고, 이를 기초로, 사전처리된 입력 오디오 정보(110a)를 제공하도록 구성된 사전처리기(120)를 택일적으로 포함한다. 오디오 인코더(100)는 또한 신호 컨버터라고도 칭해지는 에너지 압축 시간 영역-대-주파수 영역 신호 변환기(130)를 포함한다. 신호 컨버터(130)는 입력 오디오 정보(110, 110a)를 수신하고, 이를 기초로, 바람직하게는 스펙트럼 값들의 세트의 형태를 취하는 주파수 영역 오디오 정보(132)를 제공하도록 구성된다. 예를 들어, 신호 변환기(130)는 입력 오디오 정보(110, 110a)의 프레임(예컨대, 시간 영역 샘플들의 블록)을 수신하고, 각각의 오디오 프레임의 오디오 콘텐츠를 표현한 스펙트럼 값들의 세트를 제공하도록 구성될 수 있다. 게다가, 신호 변환기(130)는 입력 오디오 정보(110, 110a)의 후속하는, 오버랩하거나 또는 오버랩하지 않는, 복수의 오디오 프레임들을 수신하고, 이를 기초로, 스펙트럼 값들의 후속하는 세트들(스펙트럼 값들의 하나의 세트는 각 프레임과 연관됨)의 시퀀스를 포함하는 시간-주파수 영역 오디오 표현을 제공하도록 구성될 수 있다.

[0044] 에너지 압축 시간 영역-대-주파수 영역 신호 변환기(130)는 상이한, 오버랩하거나 또는 오버랩하지 않는 주파수 범위들과 연관된 스펙트럼 값들을 제공하는 에너지 압축 필터뱅크를 포함할 수 있다. 예를 들어, 신호 변환기(130)는 변환 윈도우를 이용하여 입력 오디오 정보(110, 110a)(또는 이것의 프레임)을 윈도우잉하고 윈도우잉된 입력 오디오 정보(110, 110a)(또는 이것의 윈도우잉된 프레임)의 변형 이산 코사인 변환을 수행하도록 구성된 윈도우잉 MDCT 변환기(130a)를 포함할 수 있다. 따라서, 주파수 영역 오디오 표현(132)은 입력 오디오 정보의 프레임과 연관된 MDCT 계수들의 형태의, 예컨대 1024개의 스펙트럼 값들의 세트를 포함할 수 있다.

[0045] 오디오 인코더(100)는 택일적으로, 주파수 영역 오디오 표현(132)을 수신하고, 이를 기초로, 후처리된 주파수 영역 오디오 표현(142)을 제공하도록 구성된 스펙트럼 후처리기(140)를 더 포함할 수 있다. 스펙트럼 후처리기(140)는 예컨대, 일시적 노이즈 셰이핑 및/또는 장기간 예측 및/또는 본 발명분야에서 알려진 임의의 다른 스펙트럼 후처리를 수행하도록 구성될 수 있다. 오디오 인코더는 택일적으로, 주파수 영역 오디오 표현(132) 또는 이것의 후처리된 버전(142)을 수신하고, 스케일링되고 양자화된 주파수 영역 오디오 표현(152)을 제공하도록 구성된 스케일러/양자화기(150)를 더 포함할 수 있다.

[0046] 오디오 인코더(100)는 택일적으로, 입력 오디오 정보(110)(또는 이것의 후처리된 버전(110a))을 수신하고, 이를 기초로, 에너지 압축 시간 영역-대-주파수 영역 신호 변환기(130)의 제어, 택일적인 스펙트럼 후처리기(140)의 제어, 및/또는 택일적인 스케일러/양자화기(150)의 제어를 위해 이용될 수 있는, 택일적인 제어 정보를 제공하도록 구성된 심리음향적 모델 처리기(160)를 더 포함한다. 예를 들어, 심리음향적 모델 처리기(160)는 입력 오

디오 정보를 분석하고, 이 입력 오디오 정보(110, 110a)의 어느 성분들이 오디오 콘텐츠의 인간 지각에 특히 중요한지와 입력 오디오 정보(110, 110a)의 어느 성분들이 오디오 콘텐츠의 인간 지각에 덜 중요한지를 결정하도록 구성될 수 있다. 따라서, 심리음향적 모델 처리기(160)는 스케일러/양자화기(150) 및/또는 스케일러/양자화기(150)에 의해 적용된 양자화 분해능에 의한 주파수 영역 오디오 표현(132, 142)의 스케일링을 조정하기 위해 오디오 인코더(100)에 의해 이용된 제어 정보를 제공할 수 있다. 결과적으로, 지각적으로 중요한 스케일 인자 대역들(즉, 오디오 콘텐츠의 인간 지각에 특별히 중요한 인접한 스펙트럼 값들의 그룹들)은 큰 스케일링 인자로 스케일링되고 상대적으로 높은 분해능으로 양자화되는 반면에, 지각적으로 덜 중요한 스케일 인자 대역들(즉, 인접한 스펙트럼 값들의 그룹들)은 상대적으로 작은 스케일링 인자로 스케일링되고 상대적으로 낮은 양자화 분해능으로 양자화된다. 따라서, 지각적으로 보다 중요한 주파수들의 스케일링된 스펙트럼 값들은 일반적으로 지각적으로 덜 중요한 주파수들의 스펙트럼 값들보다 상당히 크다.

[0047] 오디오 인코더는 또한 주파수 영역 오디오 표현(132)의 스케일링되고 양자화된 버전(152)(또는, 대안적으로, 주파수 영역 오디오 표현(132)의 후처리된 버전(142), 또는 심지어 주파수 영역 오디오 표현(132) 그 자체)를 수신하고, 산술 코드워드 정보가 주파수 영역 오디오 표현(152)을 표현하도록, 이를 기초로 산술 코드워드 정보(172a)를 제공하도록 구성된 산술 인코더(170)를 포함한다.

[0048] 오디오 인코더(100)는 또한 산술 코드워드 정보(172a)를 수신하도록 구성된 비트스트림 페이로드 포맷터(190)를 포함한다. 비트스트림 페이로드 포맷터(190)는 또한 일반적으로 추가적인 정보, 예컨대 스케일러/양자화기(150)에 의해 어느 스케일 인자들이 적용되었는지를 기술하는 스케일 인자 정보를 수신하도록 구성된다. 또한, 비트스트림 페이로드 포맷터(190)는 다른 제어 정보를 수신하도록 구성될 수 있다. 비트스트림 페이로드 포맷터(190)는 후술될 희망하는 비트스트림 구문에 따라 비트스트림을 조립함으로써 수신된 정보에 기초하여 비트스트림(112)을 제공하도록 구성된다.

[0049] 이후에는, 산술 인코더(170)에 관한 세부사항들을 설명할 것이다. 산술 인코더(170)는 주파수 영역 오디오 표현(132)의 후처리되고 스케일링되고 양자화된 복수의 스펙트럼 값들을 수신하도록 구성된다. 산술 인코더는 스펙트럼 값으로부터 최상위 비트플레인 m을 추출하도록 구성된 최상위 비트플레인 추출기(174)를 포함한다. 여기서 최상위 비트플레인은 스펙트럼 값의 최상위 비트들인 하나 또는 그 이상의 비트들(예컨대, 두 개 또는 세 개의 비트들)을 포함할 수 있다는 것을 유념해야 한다. 따라서, 최상위 비트플레인 추출기(174)는 스펙트럼 값의 최상위 비트플레인 값(176)을 제공한다.

[0050] 산술 인코더(170)는 또한 최상위 비트플레인 값 m을 표현하는 산술 코드워드 acod_m [pki][m]을 결정하도록 구성된 제1 코드워드 결정기(180)를 포함한다. 택일적으로, 코드워드 결정기(180)는 또한 예컨대, 얼마나 많은 하위 비트플레인들이 이용가능한지를 표시하는(그리고, 결과적으로, 최상위 비트플레인의 수치적 가중을 표시하는) 하나 이상의 탈출 코드워드들(이것은 또한 “ARITH_ESCAPE”으로 여기서 칭해진다)을 제공할 수 있다. 제1 코드워드 결정기(180)는 누적 도수 테이블 인덱스 pki를 갖는(또는 이것에 의해 참조되는) 선택된 누적 도수 테이블을 이용하여 최상위 비트플레인 값 m과 연관된 코드워드를 제공하도록 구성될 수 있다.

[0051] 어느 누적 도수 테이블이 선택되어야 하는지를 결정하기 위해, 산술 인코더는 바람직하게는 예컨대 어느 스펙트럼 값들이 이전에 인코딩되었는지를 관찰함으로써 산술 인코더의 상태를 추적하도록 구성된 상태 추적기(182)를 포함한다. 상태 추적기(182)는 결과적으로 상태 정보(184), 예컨대 “s” 또는 “t”로 칭해진 상태 값을 제공한다. 산술 인코더(170)는 또한 상태 정보(184)를 수신하고, 선택된 누적 도수 테이블을 기술하는 정보(188)를 코드워드 결정기(180)에게 제공하도록 구성된 누적 도수 테이블 선택기(186)를 포함한다. 예를 들어, 누적 도수 테이블 선택기(186)는 64개의 누적 도수 테이블들의 세트 중에서 어느 누적 도수 테이블이 코드워드 결정기에 의한 이용을 위해 선택되는지를 기술하는 누적 도수 테이블 인덱스 "pki"를 제공할 수 있다. 대안적으로, 누적 도수 테이블 선택기(186)는 선택된 전체 누적 도수 테이블을 코드워드 결정기에 제공할 수 있다. 따라서, 최상위 비트플레인 값 m 을 인코딩하는 실제 코드워드 acod_m[pki][m] 가 m의 값과 누적 도수 테이블 인덱스 pki에 의존하고, 결과적으로 현재 상태 정보(184)에 의존하도록, 코드워드 결정기(180)는 최상위 비트플레인 값 m 의 코드워드 acod_m[pki][m] 의 제공을 위해 선택된 누적 도수 테이블을 이용할 수 있다. 코딩 처리 및 획득된 코드워드 포맷에 관한 보다 자세한 내용은 아래에서 설명할 것이다.

[0052] 산술 인코더(170)는, 인코딩될 스펙트럼 값들 중 하나 이상의 스펙트럼 값들이 최상위 비트 플레인만을 이용하여 인코딩가능한 값들의 범위를 초과한 경우, 스케일링되고 양자화된 주파수 영역 오디오 표현(152)으로부터 하나 이상의 하위 비트플레인들을 추출하도록 구성된 하위 비트플레인 추출기(189a)를 더 포함한다. 하위 비트플레인들은 희망하는 바에 따라, 하나 이상의 비트들을 포함할 수 있다. 따라서, 하위 비트플레인 추출기(189a)는

하위 비트플레인 정보(189b)를 제공한다. 산술 인코더(170)는 또한 하위 비트플레인 정보(189d)를 수신하고, 이에 기초하여, 0개, 1개 또는 그 이상의 하위 비트플레인들의 콘텐츠를 표현하는 0개, 1개 또는 그 이상의 코드워드들 "acod_r"을 제공하도록 구성된 제2 코드워드 결정기(189c)를 포함한다. 제2 코드워드 결정기(189c)는 하위 비트플레인 정보(189b)로부터 하위 비트플레인 코드워드들 "acod_r"을 유도하기 위해 산술 인코딩 알고리즘 또는 임의의 다른 인코딩 알고리즘을 적용하도록 구성될 수 있다.

[0053] 여기서, 인코딩될 스케일링되고 양자화된 스펙트럼 값들이 상대적으로 작은 경우에 어떠한 하위 비트플레인도 존재하지 않도록 하고, 인코딩될 현재 스케일링되고 양자화된 스펙트럼 값이 중간 범위에 있는 경우 하나의 하위 비트플레인이 존재하도록 하며, 인코딩될 스케일링되고 양자화된 스펙트럼 값이 상대적으로 큰 값을 취하는 경우 하나 보다 많은 하위 비트플레인이 존재하도록, 하위 비트플레인들의 갯수는 스케일링되고 양자화된 스펙트럼 값들(152)의 값에 의존하여 달라질 수 있다는 것을 유념해야 한다.

[0054] 상기의 내용을 요약하자면, 산술 인코더(170)는 정보(152)에 의해 기술되는 스케일링되고 양자화된 스펙트럼 값들을 계층적 인코딩 처리를 이용하여 인코딩하도록 구성된다. (예컨대, 스펙트럼 값 당 하나, 두 개, 또는 세 개의 비트들을 포함한) 최상위 비트플레인은 최상위 비트플레인 값의 산술 코드워드 "acod_m[pki][m]" 을 획득하도록 인코딩된다. 하나 이상의 하위 비트플레인들(각각의 하위 비트플레인들은 예컨대 하나, 두 개 또는 세 개의 비트들을 포함한다)은 하나 이상의 코드워드들 "acod_r" 을 획득하도록 인코딩된다. 최상위 비트플레인을 인코딩할 때, 최상위 비트플레인의 값 m 은 코드워드 acod_m[pki][m] 에 맵핑된다. 이를 위해, 산술 인코더(170)의 상태에 의존하여, 즉 이전에 인코딩된 스펙트럼 값들에 의존하여 값 m 의 인코딩을 위해 64개의 상이한 누적 도수 테이블들이 이용가능하다. 따라서, 코드워드 "acod_m[pki][m]" 이 획득된다. 또한, 하나 이상의 하위 비트플레인들이 존재하는 경우 하나 이상의 코드워드들 "acod_r" 이 제공되고 비트스트림내에 포함된다.

[0055] 재설정 설명

[0056] 오디오 인코더(100)는 택일적으로, 예컨대 상태 인덱스를 디폴트 값으로 설정함으로써 콘텍스트를 재설정하여 비트레이트에서의 개선이 획득될 수 있는지 여부를 결정하도록 구성될 수 있다. 따라서, 오디오 인코더(100)는 산술 인코딩을 위한 콘텍스트가 재설정되는지를 표시하고, 또한 대응하는 디코더에서의 산술 디코딩을 위한 콘텍스트가 재설정되어야 하는지를 표시하는 재설정 정보(예컨대, "arith_reset_flag" 으로 칭해짐)를 제공하도록 구성될 수 있다.

[0057] 비트스트림 포맷과 적용된 누적 도수 테이블들에 관한 자세한 내용은 아래에서 설명될 것이다.

[0058] 4. 오디오 디코더

[0059] 이하에서는, 본 발명의 실시예에 따른 오디오 디코더를 설명할 것이다. 도 2는 이러한 오디오 디코더(200)의 개략적인 블록도를 도시한다.

[0060] 오디오 디코더(200)는 인코딩된 오디오 정보를 표현하고, 오디오 인코더(100)에 의해 제공된 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.

[0061] 오디오 디코더(200)는 비트스트림(210)을 수신하며 비트스트림(210)으로부터 인코딩된 주파수 영역 오디오 표현(222)을 추출하도록 구성된 택일적인 비트스트림 페이로드 디포맷터(220)를 포함한다. 예를 들어, 비트스트림 페이로드 디포맷터(220)는 비트스트림(210)으로부터 산술적으로 코딩된 스펙트럼 데이터, 예컨대 주파수 영역 오디오 표현의 스펙트럼 값 a 의 최상위 비트플레인 값 m 을 표현하는 산술 코드워드 "acod_m [pki][m]", 및 상기 스펙트럼 값 a 의 하위 비트플레인의 콘텐츠를 표현하는 코드워드 "acod_r" 를 추출하도록 구성될 수 있다. 따라서, 인코딩된 주파수 영역 오디오 표현(222)은 스펙트럼 값들의 산술적으로 인코딩된 표현을 구성(또는 포함)한다. 비트스트림 페이로드 디포맷터(220)는 또한 비트스트림으로부터 도 2에서는 도시되지 않은 추가적인 제어 정보를 추출하도록 구성된다. 또한, 비트스트림 페이로드 디포맷터는 택일적으로, 비트스트림(210)으로부터 상태 재설정 정보(224)(이것은 또한 산술 재설정 플래그 또는 "arith_reset_flag" 로서 칭해진다)를 추출하도록 구성된다.

[0062] 오디오 디코더(200)는 "스펙트럼 무잡음 디코더" 로서도 칭해지는 산술 디코더(230)를 포함한다. 산술 디코더(230)는 인코딩된 주파수 영역 오디오 표현(220) 및 택일적으로 상태 재설정 정보(224)를 수신하도록 구성된다. 산술 디코더(230)는 또한 스펙트럼 값들의 디코딩된 표현을 포함할 수 있는 디코딩된 주파수 영역 오디오 표현(232)을 제공하도록 구성된다. 예를 들어, 디코딩된 주파수 영역 오디오 표현(232)은 인코딩된 주파수 영역 오

디오 표현(220)에 의해 기술된 스펙트럼 값들의 디코딩된 표현을 포함할 수 있다.

[0063] 오디오 디코더(200)는 또한, 디코딩된 주파수 영역 오디오 표현(232)을 수신하고, 이를 기초로, 역으로 양자화되고 리스케일링된 주파수 영역 오디오 표현(242)을 제공하도록 구성된 택일적인 역 양자화기/리스케일러(240)를 포함한다.

[0064] 오디오 디코더(200)는 역으로 양자화되고 리스케일링된 주파수 영역 오디오 표현(242)을 수신하고, 이를 기초로, 역으로 양자화되고 리스케일링된 주파수 영역 오디오 표현(242)의 사전처리된 버전(252)을 제공하도록 구성된 택일적인 스펙트럼 사전처리기(250)를 더 포함한다. 오디오 디코더(200)는 또한 "신호 컨버터"라고도 칭해지는 주파수 영역-대-시간 영역 신호 변환기(260)를 포함한다. 신호 변환기(260)는 역으로 양자화되고 리스케일링된 주파수 영역 오디오 표현(242)(또는, 대안적으로, 역으로 양자화되고 리스케일링된 주파수 영역 오디오 표현(242) 또는 디코딩된 주파수 영역 오디오 표현(232))의 사전처리된 버전(252)을 수신하고, 이를 기초로 오디오 정보의 시간 영역 표현(262)을 제공하도록 구성된다. 주파수 영역-대-시간 영역 신호 변환기(260)는, 예컨대, 역 변형 이산 코사인 변환(inverse modified discrete cosine transform; IMDCT) 및 적절한 윈도우잉(뿐만이 아니라, 예컨대 오버랩 합산과 같은 다른 보조적 기능들)을 수행하기 위한 변환기를 포함할 수 있다.

[0065] 오디오 디코더(200)는 오디오 정보의 시간 영역 표현(262)을 수신하고, 시간 영역 후처리를 이용하여 디코딩된 오디오 정보(212)를 획득하도록 구성된 택일적인 시간 영역 후처리기(270)를 더 포함할 수 있다. 하지만, 후처리가 생략되는 경우, 시간 영역 표현(262)은 디코딩된 오디오 정보(212)와 동일할 수 있다.

[0066] 여기서 역 양자화기/리스케일러(240), 스펙트럼 사전처리기(250), 주파수 영역-대-시간 영역 신호 변환기(260) 및 시간 영역 후처리기(270)는 비트스트림 페이로드 디포맷터(220)에 의해 비트스트림(210)으로부터 추출되는 제어 정보에 의존하여 제어될 수 있다는 것을 유념해야 한다.

[0067] 오디오 디코더(200)의 전체적인 기능을 요약하자면, 디코딩된 주파수 영역 오디오 표현(232), 예컨대 인코딩된 오디오 정보의 오디오 프레임과 연관된 스펙트럼 값들의 세트는 산술 디코더(230)를 이용하여 인코딩된 주파수 영역 표현(222)에 기초하여 획득될 수 있다. 후속하여, 예컨대, MDCT 계수들일 수 있는 1024개의 스펙트럼 값들의 세트가 역으로 양자화되고, 리스케일링되며 사전처리된다. 따라서, 역으로 양자화되고, 리스케일링되며 스펙트럼적으로 사전처리된 스펙트럼 값들의 세트(예컨대, 1024개의 MDCT 계수들)가 획득된다. 그 후, 오디오 프레임의 시간 영역 표현이 역으로 양자화되고, 리스케일링되며 스펙트럼적으로 사전처리된 주파수 영역 값들의 세트(예컨대, MDCT 계수들)로부터 유도된다. 따라서, 오디오 프레임의 시간 영역 표현이 획득된다. 주어진 오디오 프레임의 시간 영역 표현은 이전의 및/또는 후속하는 오디오 프레임들의 시간 영역 표현들과 결합될 수 있다. 예를 들어, 인접한 오디오 프레임들의 시간 영역 표현들간의 천이들을 부드럽게 하고 얼라이어싱 소거를 획득하기 위해, 후속하는 오디오 프레임들의 시간 영역 표현들간의 오버랩 합산이 수행될 수 있다. 디코딩된 시간 주파수 영역 오디오 표현(232)에 기초한 디코딩된 오디오 정보(212)의 재구축에 관한 상세설명에 대해서는, 예컨대 이러한 상세한 설명이 주어져 있는 국제 표준 ISO/IEC 14496-3, 파트 3, 서브파트 4를 참조바란다. 하지만, 또 다른 정교한 오버랩 및 얼라이어싱 소거 방식들이 이용될 수 있다.

[0068] 이후에는, 산술 디코더(230)에 관한 몇가지 세부사항들을 설명할 것이다. 산술 디코더(230)는 최상위 비트플레인 값 m을 기술하는 산술 코드워드 acod_m [pki][m]을 수신하도록 구성된 최상위 비트플레인 결정기(284)를 포함한다. 최상위 비트플레인 결정기(284)는 산술 코드워드 “acod_m [pki][m]” 로부터 최상위 비트플레인 값 m을 유도하기 위해 64개의 복수의 누적 도수 테이블들을 포함한 세트 중에서의 누적 도수 테이블을 이용하도록 구성될 수 있다.

[0069] 최상위 비트플레인 결정기(284)는 코드워드 acod_m 에 기초하여 스펙트럼 값들의 최상위 비트플레인의 값들(286)을 유도하도록 구성된다. 산술 디코더(230)는 스펙트럼 값의 하나 이상의 하위 비트플레인들을 표현하는 하나 이상의 코드워드들 “acod_r”을 수신하도록 구성된 하위 비트플레인 결정기(288)를 더 포함한다. 따라서, 하위 비트플레인 결정기(288)는 하나 이상의 하위 비트플레인들의 디코딩된 값(290)을 제공하도록 구성된다. 오디오 디코더(200)는 또한 그러한 하위 비트플레인들이 현재 스펙트럼 값들에 대해 이용가능한 경우 스펙트럼 값들의 하나 이상의 하위 비트플레인들의 디코딩된 값들(290)과 스펙트럼 값들의 최상위 비트플레인의 디코딩된 값들(286)을 수신하도록 구성된 비트플레인 결합기(292)를 포함한다. 따라서, 비트플레인 결합기(292)는 디코딩된 주파수 영역 오디오 표현(232)의 일부인 디코딩된 스펙트럼 값들을 제공한다. 당연히, 산술 디코더(230)는 일반적으로 오디오 콘텐츠의 현재 프레임과 연관된 디코딩된 스펙트럼 값들의 완전 세트를 획득하기 위해 복수의 스펙트럼 값들을 제공하도록 구성된다.

[0070] 산술 디코더(230)는 산술 디코더의 상태를 기술하는 상태 인덱스(298)에 의존하여 64개의 누적 도수 테이블들 중 하나의 테이블을 선택하도록 구성된 누적 도수 테이블 선택기(296)를 더 포함한다. 산술 디코더(230)는 이전에 디코딩된 스펙트럼 값에 의존하여 산술 디코더의 상태를 추적하도록 구성된 상태 추적기(299)를 더 포함한다. 상태 정보는 택일적으로 상태 재설정 정보(224)에 응답하여 디폴트 상태 정보로 재설정될 수 있다. 따라서, 누적 도수 테이블 선택기(296)는 코드워드 "acod_m" 에 의존하여 최상위 비트플레인 값 m의 디코딩에서의 적용을 위해, 선택된 누적 도수 테이블의 인덱스(예컨대, pki), 또는 선택된 누적 도수 테이블 그 자체를 제공하도록 구성된다.

[0071] 오디오 디코더(200)의 기능을 요약하자면, 오디오 디코더(200)는 비트레이트 효율적으로 인코딩된 주파수 영역 오디오 표현(222)을 수신하고 이를 기초로 디코딩된 주파수 영역 오디오 표현을 획득하도록 구성된다. 인코딩된 주파수 영역 오디오 표현(222)에 기초하여 디코딩된 주파수 영역 오디오 표현(232)을 획득하기 위해 이용된 산술 디코더(230)에서, 인접한 스펙트럼 값들의 최상위 비트플레인의 값들의 상이한 조합들의 확률이 누적 도수 테이블을 적용하도록 구성된 산술 디코더(280)를 이용함으로써 활용된다. 다시 말하면, 스펙트럼 값들간의 통계적 의존성들은 이전에 계산되고 디코딩된 스펙트럼 값들을 관찰함으로써 획득된, 상태 인덱스(298)에 의존하여 64개의 상이한 누적 도수 테이블들을 포함한 세트 중에서 상이한 누적 도수 테이블들을 선택함으로써 활용된다.

[0072] 5. 스펙트럼 무잡음 코딩의 툴에 관한 개관

[0073] 이하에서는, 예컨대 산술 인코더(170)와 산술 디코더(230)에 의해 수행되는 인코딩 및 디코딩 알고리즘에 관한 상세사항을 설명할 것이다.

[0074] 디코딩 알고리즘의 설명에 촛점을 둔다. 하지만, 대응하는 인코딩 알고리즘은 맵핑들이 반대로 되어 있는 디코딩 알고리즘의 교시에 따라 수행될 수 있다는 것을 유념해야 한다.

[0075] 이하에서 논의할 디코딩은 일반적으로 후처리되고, 스케일링되며 양자화된 스펙트럼 값들의 소위 말하는 "스펙트럼 무잡음 코딩"을 허용하기 위해 이용된다는 점을 유념해야 한다. 스펙트럼 무잡음 코딩은 예컨대 에너지 압축 시간 영역-대-주파수 영역 변환기에 의해 획득된, 양자화된 스펙트럼의 리던던시를 한층 감소시키기 위해 오디오 인코딩/디코딩 개념에서 이용된다.

[0076] 본 발명의 실시예들에서 이용된, 스펙트럼 무잡음 코딩 방식은 동적으로 조정된 콘텍스트와 함께 산술 코딩에 기초한다. 무잡음 코딩은 양자화된 스펙트럼 값들(의 원래 표현 또는 인코딩된 표현)이 제공되고, 예컨대 이전에 디코딩된 이웃하는 복수의 스펙트럼 값들로부터 유도된 콘텍스트 의존적 누적 도수 테이블들을 이용한다. 여기서, 도 4에서는 시간 및 주파수상의 인접성이 고려된다. 그런 후 (이하에서 설명될) 누적 도수 테이블들은 가변 길이 바이너리 코드를 생성하기 위해 산술 코더에 의해 이용되고, 가변 길이 바이너리 코드로부터 디코딩된 값들을 유도하기 위해 산술 디코더에 의해 이용된다.

[0077] 예를 들어, 산술 코더(170)는 각각의 확률들에 의존하여 주어진 심볼들의 세트에 대한 바이너리 코드를 생성한다. 바이너리 코드는 심볼 세트가 놓여 있는 확률 구간을 코드워드에 맵핑함으로써 생성된다.

[0078] 이하에서는, 스펙트럼 무잡음 코딩의 툴의 또 다른 간단한 개관이 주어질 것이다. 스펙트럼 무잡음 코딩은 양자화된 스펙트럼의 리던던시를 한층 감소시키는데 이용된다. 스펙트럼 무잡음 코딩 방식은 동적으로 조정된 콘텍스트와 함께 산술 코딩에 기초한다. 무잡음 코딩은 양자화된 스펙트럼 값들이 제공되고, 예컨대 이전에 디코딩된 이웃하는 일곱 개의 스펙트럼 값들로부터 유도된 콘텍스트 의존적 누적 도수 테이블들을 이용한다.

[0079] 여기서, 도 4에서는 시간 및 주파수상의 인접성이 고려된다. 그런 후 누적 도수 테이블들은 가변 길이 바이너리 코드를 생성하기 위해 산술 코더에 의해 이용된다.

[0080] 산술 코더는 주어진 심볼들의 세트에 대한 바이너리 코드와 이들 각각의 확률을 생성한다. 바이너리 코드는 심볼들의 세트가 놓여 있는 확률 구간을 코드워드에 맵핑함으로써 생성된다.

[0081] 6. 디코딩 처리

[0082] 6.1. 디코딩 처리 개관

[0083] 이하에서는, 복수의 스펙트럼 값들을 디코딩하는 처리의 의사 프로그램 코드 표현을 도시하는, 스펙트럼 값을 디코딩하는 처리의 개관이 도 3을 참조하여 주어질 것이다.

[0084] 복수의 스펙트럼 값들을 디코딩하는 처리는 콘텍스트의 초기화(310)를 포함한다. 콘텍스트의 초기화(310)는 함수 "arith_map_context (lg)"를 이용한 이전 콘텍스트로부터의 현재 콘텍스트의 유도를 포함한다. 이전 콘텍

스트로부터의 현재 콘텍스트의 유도는 콘텍스트의 재설정을 포함할 수 있다. 이하에서는 이전 콘텍스트로부터의 현재 콘텍스트의 유도 및 콘텍스트의 재설정 모두를 설명할 것이다.

[0085] 복수의 스펙트럼 값들의 디코딩은 또한 스펙트럼 값 디코딩(312)과 콘텍스트 업데이트(314)의 반복을 포함하며, 콘텍스트 업데이트는 아래에서 설명되는 함수 “Arith_update_context(a,i,lg)” 에 의해 수행된다. 스펙트럼 값 디코딩(312) 및 콘텍스트 업데이트(314)는 lg회 반복되며, 여기서 lg는 디코딩될 스펙트럼 값들(예컨대, 오디오 프레임)의 갯수를 표시한다. 스펙트럼 값 디코딩(312)은 콘텍스트 값 계산(312a), 최상위 비트플레인 디코딩(312b), 및 하위 비트플레인 가산(312c)을 포함한다.

[0086] 상태 값 계산(312a)은 제1 상태값 s 를 반환하는 함수 “arith_get_context(i, lg, arith_reset_flag, N/2)” 를 이용한 제1 상태값 s 의 계산을 포함한다. 상태값 계산(312a)은 레벨 값 "lev0" 및 레벨값 "lev"의 계산을 포함하며, 이 레벨 값들 "lev0", "lev" 은 제1 상태값 s 를 우측으로 24비트만큼 쉬프트시킴으로써 획득된다. 상태 값 계산(312a)은 또한 참조번호 312a로서 도 3에서 도시된 공식에 따른 제2 상태 값 t 의 계산을 포함한다.

[0087] 최상위 비트플레인 디코딩(312b)은 디코딩 알고리즘(312ba)의 반복적인 실행을 포함하며, 변수 j 는 알고리즘(312ba)의 첫번째 실행 전에 0으로 초기화된다.

[0088] 알고리즘(312ba)은 후술되는 함수 “arith_get_pk()” 를 이용하고 제2 상태 값 t 에 의존하고 또한 레벨값들 "lev" 및 "lev0"에 의존한 상태 인덱스 "pki" (이것은 또한 누적 도수 테이블 인덱스로서 역할을 할 수 있음)의 계산을 포함한다. 알고리즘(312ba)은 또한 상태 인덱스 pki 에 의존한 누적 도수 테이블의 선택을 포함하며, 변수 “cum_freq” 는 상태 인덱스 pki 에 의존하여 64개의 누적 도수 테이블들 중에서 하나의 테이블의 시작 어드레스로 설정될 수 있다. 또한, 변수 “cfl” 는 예컨대 알파벳 심볼들의 갯수, 즉 디코딩될 수 있는 상이한 값들의 갯수와 동일한 선택된 누적 도수 테이블의 길이로 초기화될 수 있다. 여덟 개의 상이한 최상위 비트플레인 값들과 탈출 심볼이 디코딩될 수 있으므로, 최상위 비트플레인 값 m 의 디코딩을 위해 이용가능한 “arith_cf_m[pki=0][9]”에서부터 “arith_cf_m[pki=63][9]”까지의 모든 누적 도수 테이블들의 길이는 9이다. 후속하여, 최상위 비트플레인 값 m 은 (변수 “cum_freq” 와 변수 “cfl”에 의해 기술된) 선택된 누적 도수 테이블을 고려하여, 함수 “arith_decode()” 를 실행함으로써 획득될 수 있다. 최상위 비트플레인 값 m 을 유도할 때, 비트스트림(210)의 “acod_m”이라고 호칭된 비트들이 평가될 수 있다(예컨대, 도 6g를 참조).

[0089] 알고리즘(312ba)은 또한 최상위 비트 플레인 값 m 이 탈출 심볼 “ARITH_ESCAPE” 과 동일한지 아닌지 여부를 체크하는 것을 포함한다. 만약 최상위 비트플레인 값 m 이 산술 탈출 심볼과 동일하지 않는 경우, 알고리즘(312ba)은 중지되고("break" 조건"), 이에 따라 알고리즘(312ba)의 나머지 명령들은 스킵된다. 따라서, 처리의 실행은 스펙트럼 값 a 를 최상위 비트플레인 값 m(명령 “a=m”)과 동일하게 설정하는 것으로 이어진다. 이와 대조적으로, 디코딩된 최상위 비트플레인 값 m 이 산술 탈출 심볼 “ARITH_ESCAPE” 과 동일한 경우, 레벨 값 "lev" 은 1만큼 증가된다. 언급한 바와 같이, 그런 후 알고리즘(312ba)은 디코딩된 최상위 비트플레인 값 m 이 산술 탈출 심볼과 상이할 때 까지 반복된다.

[0090] 최상위 비트플레인 디코딩이 완료되자마자, 즉 산술 탈출 심볼과 상이한 최상위 비트플레인 값 m 이 디코딩되자마자, 스펙트럼 값 변수 "a" 는 최상위 비트플레인 값 m 과 동일하게 설정된다. 후속하여, 예컨대 도 3에서의 참조번호 312c에서 도시된 바와 같이 하위 비트플레인들이 획득된다. 스펙트럼 값의 각각의 하위 비트플레인에 대해, 두 개의 바이너리 값들 중 하나의 바이너리 값이 디코딩된다. 예를 들어, 하위 비트플레인 값 r 이 획득된다. 후속하여, 스펙트럼 값 변수 "a" 는 스펙트럼 값 변수 "a"의 콘텐츠를 좌측으로 1비트만큼 쉬프트시키고 현재 디코딩된 하위 비트플레인 값 r 을 최하위 비트로서 추가시킴으로써 업데이트된다. 하지만, 하위 비트플레인들의 값들을 획득하기 위한 개념은 본 발명에 대해 특별히 관련성이 있는 것은 아님을 유념해야 한다. 몇몇 실시예들에서는, 임의의 하위 비트플레인들의 디코딩은 심지어 생략될 수 있다. 대안적으로, 상이한 디코딩 알고리즘들이 이러한 목적으로 이용될 수 있다.

[0091] 6.2. 도 4에 따른 디코딩 순서

[0092] 이후에는, 스펙트럼 값들의 디코딩 순서를 설명할 것이다.

[0093] 스펙트럼 계수들은 최저 주파수 계수로부터 시작해서 최고 주파수 계수로 진행하면서 무잡음방식으로 코딩되어 (예컨대 비트스트림 내에서) 전달된다.

[0094] (예컨대, ISO/IEC 14496, 파트 3, 서브파트 4에서 논의된 변형 이산 코사인 변환을 이용하여 획득된) 진보된 오디오 코딩으로부터의 계수들은 “x_ac_quant[g][win][sfb][bin]” 이라고 칭해진 어레이에 저장되고, 어레이에

수신되어 저장된 순서로 코드워드들이 디코딩될 때, “bin” (주파수 인덱스) 이 가장 급속하게 증분하는 인덱스이고 “g” 가 가장 느리게 증분하는 인덱스이도록 무잡음 코딩 코드워드(예컨대, acod_m, acod_r)의 전달의 순서는 정해진다.

[0095] 저주파수와 연관된 스펙트럼 계수들은 고주파수와 연관된 스펙트럼 계수들에 앞서 인코딩된다.

[0096] 변환 코딩된 여기(transform coded excitation; tcx)로부터의 계수들은 어레이 x_tcx_invquant[win][bin]에 직접 저장되고, 무잡음 코딩 코드워드들이 수신되어 어레이에 저장된 순서로 디코딩될 때, “bin” 이 가장 급속하게 증분하는 인덱스이고 “win” 이 가장 느리게 증분하는 인덱스가 되도록 무잡음 코딩 코드워드의 전달 순서가 정해진다. 다시 말하면, 스펙트럼 값들이 음성 코더의 선형 예측 필터의 변환 코딩된 여기를 기술하는 경우, 스펙트럼 값들 a 는 변환 코딩된 여기의 인접하고 증분하는 주파수들과 연관된다.

[0097] 저주파수와 연관된 스펙트럼 계수들은 고주파수와 연관된 스펙트럼 계수들에 앞서 인코딩된다.

[0098] 특히, 오디오 디코더(200)는 주파수 영역-대-시간 영역 신호 변환을 이용한 시간 영역 오디오 신호 표현의 "직접적" 생성과 주파수 영역-대-시간 영역 신호 변환기의 출력에 의해 여기된 선형 예측 필터 및 주파수 영역-대-시간 영역 디코더 모두를 이용한 오디오 신호 표현의 "간접적" 제공에 대해, 산술 디코더(230)에 의해 제공된 디코딩된 주파수 영역 오디오 표현(232)을 적용하도록 구성될 수 있다.

[0099] 다시 말하면, 산술 디코더(200)(이에 대한 기능은 여기서 자세하게 설명된다)는 주파수 영역에서 인코딩된 오디오 콘텐츠의 시간 주파수 영역 표현의 스펙트럼 값들을 디코딩하고, 선형 예측 영역에서 인코딩된 음성 신호를 디코딩하도록 조정된 선형 예측 필터를 위한 자극 신호의 시간 주파수 영역 표현의 제공에 적합하다. 따라서, 산술 디코더는 주파수 영역 인코딩된 오디오 콘텐츠와 선형 예측 주파수 영역 인코딩된 오디오 콘텐츠(변환 코딩된 여기 선형 예측 영역 모드) 모두를 처리할 수 있는 오디오 디코더에서 이용하는데 적합하다.

[0100] 6.3. 도 5a 및 도 5b에 따른 콘텍스트 초기화

[0101] 이하에서는, 단계 310에서 수행되는 콘텍스트 초기화(이것은 또한 "콘텍스트 맵핑"이라고 칭해진다)를 설명할 것이다.

[0102] 콘텍스트 초기화는 도 5a에서 도시된 알고리즘 “arith_map_context()” 에 따른 현재 콘텍스트와 과거 콘텍스트간의 맵핑을 포함한다. 살펴볼 수 있는 바와 같이, 현재 콘텍스트는 2의 제1 차원과 n_context의 제2 차원을 갖는 어레이의 형태를 취하는 글로벌 변수 q[2][n_context] 에 저장된다. 과거 콘텍스트는 n_context의 차원을 갖는 테이블의 형태를 취하는 변수 qs[n_context] 에 저장된다. 변수 “previous_lg” 는 과거 콘텍스트의 스펙트럼 값들의 갯수를 기술한다.

[0103] 변수 “lg” 는 프레임에서 디코딩할 스펙트럼 계수들의 갯수를 기술한다. 변수 “previous_lg” 는 이전 프레임의 스펙트럼 라인들의 이전 갯수를 기술한다.

[0104] 콘텍스트의 맵핑은 알고리즘 “arith_map_context()” 에 따라 수행될 수 있다. 여기서, i=0 내지 i=lg-1에 대하여 현재(예컨대, 주파수 영역 인코딩된) 오디오 프레임과 연관된 스펙트럼 값들의 갯수가 이전 오디오 프레임과 연관된 스펙트럼 값들의 갯수와 동일한 경우, 함수 “arith_map_context()” 는 현재 콘텍스트 어레이 q 의 엔트리들 q[0][i] 을 과거 콘텍스트 어레이 qs 의 값들 qs[i]로 설정한다는 것을 유념해야 한다.

[0105] 하지만, 현재 오디오 프레임과 연관된 스펙트럼 값들의 갯수가 이전 오디오 프레임과 연관된 스펙트럼 값들의 갯수와 상이한 경우 보다 복잡한 맵핑이 수행된다. 하지만, 이 경우에서의 맵핑에 관한 상세사항은 본 발명의 핵심적인 아이디어와는 특별히 관련있지 않으며, 도 5a의 의사 프로그램 코드에 대해서 자세하게 참조한다.

[0106] 6.4. 도 5b 및 도 5c에 따른 상태 값 계산

[0107] 이하에서는, 상태 값 계산(312a)을 보다 자세하게 설명할 것이다.

[0108] (도 3에서 도시된) 제1 상태 값 s 는 도 5b와 도 5c에서 도시된 의사 프로그램 코드 표현인 함수 “arith_get_context(i, lg, arith_reset_flag, N/2)” 의 반환 값으로서 획득될 수 있다는 것을 유념해야 한다.

[0109] 상태 값의 계산과 관련하여, 상태 평가를 위해 이용된 콘텍스트를 도시한 도 4를 또한 참조한다. 도 4는 시간과 주파수상에서의 스펙트럼 값들의 2차원 표현을 도시한다. 가로좌표(410)는 시간을 기술하고, 세로좌표(412)는 주파수를 기술한다. 도 4에서 살펴볼 수 있는 바와 같이, 디코딩할 스펙트럼 값(420)은 시간 인덱스 t0 및 주파수 인덱스 i와 연관된다. 살펴볼 수 있는 바와 같이, 시간 인덱스 t0에서, 주파수 인덱스들 i-1, i-2 및 i-3

을 갖는 튜플(tuple)들은 주파수 인덱스 i 를 갖는 스펙트럼 값(420)이 디코딩되는 시간에서는 이미 디코딩되어 있다. 도 4로부터 살펴볼 수 있는 바와 같이, 시간 인덱스 t0와 주파수 인덱스 i-1을 갖는 스펙트럼 값(430)은 스펙트럼 값(420)이 디코딩되기 전에 이미 디코딩되어 있으며, 스펙트럼 값(430)은 스펙트럼 값(420)의 디코딩에 이용된 콘텍스트용으로 고려된다. 마찬가지로, 시간 인덱스 t0와 주파수 인덱스 i-2를 갖는 스펙트럼 값(434)은 스펙트럼 값(420)이 디코딩되기 전에 이미 디코딩되었으며, 스펙트럼 값(434)은 스펙트럼 값(420)의 디코딩에 이용된 콘텍스트용으로 고려된다. 마찬가지로, 시간 인덱스 t-1과 주파수 인덱스 i-2를 갖는 스펙트럼 값(440), 시간 인덱스 t-1과 주파수 인덱스 i-1을 갖는 스펙트럼 값(444), 시간 인덱스 t-1과 주파수 인덱스 i를 갖는 스펙트럼 값(448), 시간 인덱스 t-1과 주파수 인덱스 i+1를 갖는 스펙트럼 값(452) 및 시간 인덱스 t-1과 주파수 인덱스 i+2를 갖는 스펙트럼 값(456)은 스펙트럼 값(420)이 디코딩되기 전에 이미 디코딩되었으며, 스펙트럼 값(420)을 디코딩하기 위해 이용된 콘텍스트의 결정을 위한 것으로 고려된다. 스펙트럼 값(420)이 디코딩되고 콘텍스트용으로 고려될 때에 이미 디코딩된 스펙트럼 값들(계수들)은 음영처리된 정사각형으로 도시된다. 이와 대조적으로, 점선을 갖는 정사각형들로 표현된, (스펙트럼 값(420)이 디코딩될 때) 이미 디코딩된 몇몇의 다른 스펙트럼 값들과, 점선을 갖는 원들로 도시된 (스펙트럼 값(420)이 디코딩될 때) 아직 디코딩되지 않은 다른 스펙트럼 값들은 스펙트럼 값(420)을 디코딩하기 위한 콘텍스트를 결정하기 위해 이용되지 않는다.

[0110] 하지만, 그럼에도 불구하고 스펙트럼 값(420)을 디코딩하기 위한 콘텍스트의 "정상적인"(또는 "보통의") 계산을 위해 이용되지 않는 이러한 스펙트럼 값들 중의 몇몇은 각자의 크기에 관한 미리결정된 조건을 개별적으로 또는 다함께 충족시키는 이전에 인코딩된 인접한 복수의 스펙트럼 값들의 검출을 위해 평가될 수 있다는 것을 유념해야 한다.

[0111] 이제 의사 프로그램 코드의 형태로 함수 “arith_get_context()”의 기능을 보여주는 도 5b와 도 5c를 참조하여, 함수 “arith_get_context()”에 의해 수행되는 제1 콘텍스트 값 "s"의 계산에 관한 몇가지 보다 상세한 사항을 설명할 것이다.

[0112] 함수 “arith_get_context()”는 디코딩할 스펙트럼 값의 인덱스 i를 입력 변수들로서 수신한다는 것을 유념해야 한다. 인덱스 i는 일반적으로 주파수 인덱스이다. 입력 변수 lg 는 (현재 오디오 프레임에 대한) 기대 양자화 계수들의 (총) 갯수를 기술한다. 변수 N 은 변환 라인들의 갯수를 기술한다. 플래그 “arith_reset_flag”는 콘텍스트가 재설정되어야 하는지 여부를 표시한다. 함수 “arith_get_context” 는, 출력 값으로서, 연쇄 상태 인덱스 s와 예상 비트플레인 레벨 lev0를 표현하는 변수 "t"를 제공한다.

[0113] 함수 “arith_get_context()”는 정수 변수들 a0, c0, c1, c2, c3, c4, c5, c6, lev0, 및 “region”을 이용한다.

[0114] 함수 “arith_get_context()”는 메인 기능 블록들로서, 제1 산술 재설정 처리(510), 이전에 디코딩된 인접한 복수의 제로 스펙트럼 값들의 그룹의 검출(512), 제1 변수 설정(514), 제2 변수 설정(516), 레벨 조정(518), 영역 값 설정(520), 레벨 조정(522), 레벨 한정(524), 산술 재설정 처리(526), 제3 변수 설정(528), 제4 변수 설정(530), 제5 변수 설정(532), 레벨 조정(534), 및 선택적 반환 값 계산(536)을 포함한다.

[0115] 제1 산술 재설정 처리(510)에서, 디코딩될 스펙트럼 값의 인덱스가 제로와 동일한 동안에, 산술 재설정 플래그 “arith_reset_flag” 가 설정되는지 여부를 체크한다. 이 경우, 제로의 콘텍스트 값이 반환되고, 함수는 중지된다.

[0116] 산술 재설정 플래그가 비활성되어 있고 디코딩될 스펙트럼 값의 인덱스 i가 제로와는 상이한 경우에서만 수행되는, 이전에 디코딩된 복수의 제로 스펙트럼 값들의 그룹의 검출(512)에서, 참조번호 512a에서 도시된 바와 같이 “flag”라고 칭해진 변수는 1로 초기화되고, 참조번호 512b에서 도시된 바와 같이 평가될 스펙트럼 값의 영역이 결정된다. 후속하여, 참조번호 512b에서 도시된 바와 같이 결정되는 스펙트럼 값들의 영역은 참조번호 512c에서 도시된 바와 같이 평가된다. 이전에 디코딩된 제로 스펙트럼 값들의 충분한 영역이 존재하는 것으로 발견된 경우, 참조번호 512d에서 도시된 바와 같이 1의 콘텍스트 값이 반환된다. 예를 들어, 디코딩될 스펙트럼 값의 인덱스 i가 최대 주파수 인덱스 lg-1에 근접해 있지 않는다면, 상위 주파수 인덱스 경계 “lim_max”는 i+6으로 설정되고, 이 경우 참조번호 512b에서 도시된 바와 같이, 상위 주파수 인덱스 경계의 특정한 설정이 행해진다. 더군다나, 하위 주파수 인덱스 경계 “lim_min”는, 디코딩될 스펙트럼 값의 인덱스 i가 제로에 근접해 있지 않는다면(i+lim_min<0), -5로 설정되고, 이 경우 참조번호 512b에서 도시된 바와 같이, 하위 주파수 인덱스 경계 lim_min의 특정한 계산이 수행된다. 단계 512b에서 결정된 스펙트럼 값들의 영역을 평가할 때, 제일먼저 하위 주파수 인덱스 경계 lim_min 와 제로사이에서 네거티브 주파수 인덱스들 k에 대한 평가가 수행된다. lim_min 와 제로사이의 주파수 인덱스들 k에 대해, 콘텍스트 값들 q[0][k].c와 q[1][k].c 중에서 적어도 하나가

제로와 동일한지 여부가 확인된다. 하지만, lim_min 와 제로사이의 임의의 주파수 인덱스들 k에 대해서 콘텍스트 값들 q[0][k].c와 q[1][k].c 모두가 상이한 경우, 제로 스펙트럼 값들의 충분한 그룹이 존재하지 않다라고 결단내리고 평가(512c)는 중지된다. 후속하여, lim_max 와 제로사이의 주파수 인덱스들에 대한 콘텍스트 값들 q[0][k].c이 평가된다. lim_max 와 제로사이의 임의의 주파수 인덱스들에 대해서 콘텍스트 값들 q[0][k].c 중 임의의 값이 제로와 상이한 것으로 발견된 경우, 이전에 디코딩된 제로 스펙트럼 값들의 충분한 그룹이 존재하지 않다라고 결단내리고, 평가(512c)는 중지된다. 하지만, lim_min와 제로사이의 모든 주파수 인덱스들 k에 대해, 제로와 동일한 적어도 하나의 콘텍스트 값 q[0][k].c 또는 q[1][k].c이 존재한다라고 발견되고, 제로와 lim_max사이의 모든 주파수 인덱스 k에 대해 제로 콘텍스트 값 q[0][k].c 이 존재한 경우, 이전에 디코딩된 제로 스펙트럼 값들의 충분한 그룹이 존재한다라고 결단내린다. 따라서, 이 경우에서 이러한 조건을 표시하기 위해 어떠한 추가적인 계산 없이 1의 콘텍스트 값이 반환된다. 다시 말하면, 제로 값을 갖는 복수의 콘텍스트 값들 q[0][k].c, q[1][k].c의 충분한 그룹이 확인된 경우, 계산들(514, 516, 518, 520, 522, 524, 526, 528, 530, 532, 534, 536)은 생략된다. 다시 말하면, 미리결정된 조건이 충족되었다라는 검출에 응답하여 콘텍스트 상태 (s)를 기술하는 반환된 콘텍스트 값은 이전에 디코딩된 스펙트럼 값들로부터 독립적으로 결정된다.

[0117] 그렇지 않은 경우, 즉 제로인 콘텍스트 값들 [q][0][k].c, [q][1][k].c의 충분한 그룹이 존재하지 않는 경우에는, 계산들(514, 516, 518, 520, 522, 524,526, 528, 530, 532, 534, 536) 중 적어도 몇몇이 실행된다.

[0118] 디코딩될 스펙트럼 값의 인덱스 i가 1 미만인 경우에(및 이 경우에만) 선택적으로 실행되는 제1 변수 설정(514)에서, 변수 a0은 콘텍스트 값 q[1][i-1]을 취하도록 초기화되고, 변수 c0은 변수 a0의 절대값을 취하도록 초기화된다. 변수 "lev0"는 제로의 값을 취하도록 초기화된다. 후속하여, 변수 a0이 상대적으로 큰 절대값, 즉 -4보다 작거나 또는 4 이상인 값을 포함한 경우 변수들 "lev0" 및 c0은 증가된다. 우측 쉬프트 연산에 의해 변수 a0의 값이 -4와 3사이의 범위에 놓여 있을 때 까지, 변수들 "lev0" 및 c0의 증가는 반복적으로 수행된다(단계 514b).

[0119] 후속하여, 변수들 "lev0" 및 c0은 각각 최대값 7과 3으로 제한된다(단계 514c).

[0120] 디코딩될 스펙트럼 값의 인덱스 i가 1과 동일하고 산술 재설정 플래그(“arith_reset_flag”)가 활성화된 경우, 콘텍스트 값은 반환되고, 이것은 변수들 lev0 및 c0만을 기초로 하여 계산된다(단계 514d). 따라서, 디코딩될 스펙트럼 값과 동일한 시간 인덱스를 가지며 디코딩될 스펙트럼 값의 주파수 인덱스 i보다 1만큼 작은 주파수 인덱스를 갖는 이전에 디코딩된 단일 스펙트럼 값만이 콘텍스트 계산을 위해 고려된다(단계 514d). 그렇지 않은 경우, 즉 산술 재설정 기능이 존재하지 않는 경우, 변수 c4는 초기화된다(단계 514e).

[0121] 결론적으로, 제1 변수 설정(514)에서, 변수들 "lev0" 및 c0은 이전에 디코딩된 스펙트럼 값에 의존하여 초기화되고, 현재 디코딩되는 스펙트럼 값과 동일한 프레임 및 선행하는 스펙트럼 빈 i-1에 대해 디코딩된다. 변수 c4는 이전에 디코딩된 스펙트럼 값에 의존하여 초기화되고, 현재 디코딩되는 스펙트럼 값과 연관된 주파수보다 (예컨대 1개 주파수 빈만큼)낮은 주파수를 갖고 (시간 인덱스 t-1을 갖는) 이전 오디오 프레임에 대해 디코딩된다.

[0122] 현재 디코딩되는 스펙트럼 값의 주파수 인덱스가 1보다 큰 경우에(및 이경우에만) 선택적으로 실행되는 제2 변수 설정(516)은 변수 c1 및 c6의 초기화와, 변수 lev0의 업데이트를 포함한다. 변수 c1은 현재 디코딩되는 스펙트럼 값의 주파수보다 (예컨대 2개의 주파수 빈만큼) 작은 주파수를 갖는 현재 오디오 프레임의 이전에 디코딩된 스펙트럼 값과 연관된 콘텍스트 값 q[1][i-2].c에 의존하여 업데이트된다. 마찬가지로, 변수 c6은 현재 디코딩되는 스펙트럼 값과 연관된 주파수보다 (예컨대 2개 주파수 빈만큼) 작은 연관된 주파수를 갖는 (시간 인덱스 t-1을 갖는)이전 프레임의 이전에 디코딩된 스펙트럼 값을 기술하는 콘텍스트 값 q[0][i-2].c에 의존하여 초기화된다. 또한, 레벨 값 q[1][i-2].l이 lev0보다 큰 경우, 레벨 변수 "lev0"는 현재 디코딩되는 스펙트럼 값과 연관된 주파수보다 (예컨대 2개 주파수 빈만큼) 작은 연관된 주파수를 갖는 현재 프레임의 이전에 디코딩된 스펙트럼 값과 연관된 레벨 값 q[1][i-2].l으로 설정된다.

[0123] 디코딩될 스펙트럼 값의 인덱스 i가 2보다 큰 경우(및 이 경우에만) 레벨 조정(518) 및 영역 값 설정(520)은 선택적으로 실행된다. 레벨 조정(518)에서, 현재 디코딩되는 스펙트럼 값과 연관된 주파수보다 (예컨대 3개 주파수 빈만큼) 작은 연관 주파수를 갖는 현재 프레임의 이전에 디코딩된 스펙트럼 값과 연관된 레벨 값 q[1][i-3].l이 레벨 값 lev0보다 큰 경우, 레벨 변수 "lev0"는 q[1][i-3].l의 값으로 증가된다.

[0124] 영역 값 설정(520)에서, 변수 “region”은 복수의 스펙트럼 영역들 중에서 현재 디코딩되는 스펙트럼 값이 배열되는 스펙트럼 영역에서의 평가에 의존하여 설정된다. 예를 들어, 현재 디코딩되는 스펙트럼 값이 주파수 빈

들의 제1(가장 아래) 4분의 1 안에 있는 (주파수 빈 인덱스 i를 갖는) 주파수 빈과 연관된 것으로 발견된 경우(0 ≤ i < N/4), 영역 변수 "region"는 제로로 설정된다. 그렇지 않고, 현재 디코딩되는 스펙트럼 값이 현재 프레임과 연관된 주파수 빈들의 제2 4분의 1안에 있는 주파수 빈과 연관된 경우(N/4 ≤ i < N/2), 영역 변수는 1의 값으로 설정된다. 그렇지 않고, 현재 디코딩되는 스펙트럼 값이 주파수 빈들의 후반부(하반부) 안에 있는 주파수 빈과 연관된 경우(N/2 ≤ i < N), 영역 변수는 2로 설정된다. 따라서, 영역 변수는 현재 디코딩되는 스펙트럼 값과 연관된 주파수 영역에 대한 평가에 의존하여 설정된다. 두 개 이상의 주파수 영역들이 구별될 수 있다.

[0125] 현재 디코딩되는 스펙트럼 값이 3보다 큰 스펙트럼 인덱스를 포함한 경우(및 이 경우에만) 추가적인 레벨 조정(522)이 실행된다. 이 경우, 현재 디코딩되는 스펙트럼 값과 연관된 주파수보다 예컨대 4개 주파수 빈만큼 작은 주파수와 연관된 현재 프레임의 이전에 디코딩된 스펙트럼 값과 연관된 레벨 값 q[i][i-4].l이 현재 레벨 "lev0"보다 큰 경우, 레벨 변수 "lev0"는 증가된다(q[1][i-4].l 값으로 설정된다)(단계 522). 레벨 변수 "lev0"는 최대값 3까지 제한된다(단계 524).

[0126] 산술 재설정 조건이 검출되고 현재 디코딩되는 스펙트럼 값의 인덱스 i가 1보다 큰 경우, 변수들 c0, c1, lev0 뿐만이 아니라 영역 변수 "region"에 의존하여 상태 값이 반환된다(단계 526). 따라서, 산술 재설정 조건이 주어진 경우, 임의의 이전 프레임들의 이전에 디코딩된 스펙트럼 값들은 도외시된다.

[0127] 제3 변수 설정(528)에서, 변수 c2는 (시간 인덱스 t-1을 갖는) 이전 오디오 프레임의 이전에 디코딩된 스펙트럼 값(이전에 디코딩된 스펙트럼 값은 현재 디코딩되는 스펙트럼 값과 동일한 주파수와 연관되어 있음)과 연관된 콘텍스트 값 q[0][i].c으로 설정된다.

[0128] 제4 변수 설정(530)에서, 현재 디코딩되는 스펙트럼 값이 최고 가능 주파수 인덱스 lg-1과 연관되지 않는다면, 변수 c3는 주파수 인덱스 i+1을 갖는 이전 오디오 프레임의 이전에 디코딩된 스펙트럼 값과 연관된 콘텍스트 값 q[0][i+1].c으로 설정된다.

[0129] 제5 변수 설정(532)에서, 현재 디코딩되는 스펙트럼 값의 주파수 인덱스 i가 최대 주파수 인덱스 값과 아주 가까이 있는 경우(즉, 주파수 인덱스 값 lg-2 또는 lg-1를 취하는 경우)가 아니라면, 변수 c5는 주파수 인덱스 i+2를 갖는 이전 오디오 프레임의 이전에 디코딩된 스펙트럼 값과 연관된 콘텍스트 값 q[0][i+2].c으로 설정된다.

[0130] 주파수 인덱스 i가 제로와 동일한 경우(즉, 현재 디코딩되는 스펙트럼 값이 최저 스펙트럼 값인 경우), 레벨 변수 "lev0"의 추가적인 조정이 수행된다. 이 경우, 현재 인코딩되는 스펙트럼 값과 연관된 주파수와 비교하여, 이와 동일한 주파수 또는 심지어 이보다 높은 주파수와 연관된 이전 오디오 프레임의 이전에 디코딩된 스펙트럼 값이 상대적으로 큰 값을 취하는 것을 표시하는 값 3을 변수 c2 또는 c3가 취하는 경우, 레벨 변수 "lev0"는 제로로부터 1로 증가된다.

[0131] 선택적인 반환 값 계산(536)에서, 현재 디코딩되는 스펙트럼 값들의 인덱스 i가 제로 값, 1의 값, 또는 보다 큰 값을 취하는지 여부에 의존하여 반환 값이 계산된다. 참조번호 536a에서 나타난 바와 같이, 인덱스 i가 제로 값을 취하는 경우, 반환 값은 변수들 c2, c3, c5 및 lev0에 의존하여 계산된다. 참조번호 536b에서 도시된 바와 같이, 인덱스 i가 1의 값을 취하는 경우, 반환 값은 변수들 c0, c2, c3, c4, c5, 및 "lev0"에 의존하여 계산된다. 인덱스 i가 제로 또는 1과는 상이한 값을 취하는 경우(참조번호 536c), 반환 값은 변수들 c0, c2, c3, c4, c1, c5, c6, "region"에 의존하여 계산된다.

[0132] 위 내용을 요약해보면, 콘텍스트 값 계산 "arith_get_context()"은 이전에 디코딩된 복수의 제로 스펙트럼 값들(또는 적어도, 충분히 작은 스펙트럼 값들)의 그룹의 검출(512)을 포함한다. 이전에 디코딩된 제로 스펙트럼 값들의 충분한 그룹이 발견된 경우, 반환 값을 1로 설정함으로써 특정 콘텍스트의 존재가 표시된다. 그렇지 않은 경우, 콘텍스트 값 계산이 수행된다. 콘텍스트 값 계산에서, 이전에 디코딩된 스펙트럼 값들이 얼마나 많이 평가되어야 하는지를 결정하기 위해 인덱스 값 i가 평가된다라고 일반적으로 말할 수 있다. 예를 들어, 현재 디코딩되는 스펙트럼 값의 주파수 인덱스 i가 하위 경계(예컨대, 제로)에 근접해 있거나, 또는 상위 경계(예컨대, lg-1)에 근접해 있는 경우 평가되어진 이전에 디코딩된 스펙트럼 값들의 갯수는 감소된다. 또한, 현재 디코딩되는 스펙트럼 값의 주파수 인덱스 i가 최소값으로부터 충분히 멀리 떨어져 있는 경우라할지라도, 상이한 스펙트럼 영역들은 영역 값 설정(520)에 의해 구별된다. 따라서, 상이한 스펙트럼 영역들(예컨대, 제1의 저주파수 스펙트럼 영역, 제2의 중간 주파수 스펙트럼 영역, 및 제3의 고주파수 스펙트럼 영역)의 상이한 통계적 특성들이 고려된다. 반환된 콘텍스트 값이, 현재 디코딩되는 스펙트럼 값이 제1의 미리결정된 주파수 영역안에 있거나 또

는 제2의 미리결정된 주파수 영역(또는 임의의 다른 미리결정된 주파수 영역)안에 있는지 여부에 의존하도록, 반환 값으로서 계산된 콘텍스트 값은 변수 “region”에 의존적이다.

[0133] 6.5 맵핑 룰 선택

[0134] 이하에서는, 심볼 코드로의 코드 값의 맵핑을 기술하는 맵핑 룰, 예컨대 누적 도수 테이블의 선택을 설명할 것이다. 맵핑 룰의 선택은 상태 값 s 또는 t에 의해 기술되는 콘텍스트 상태에 의존하여 행해진다.

[0135] 6.5.1 도 5d에 따른 알고리즘을 이용한 맵핑 룰 선택

[0136] 이하에서는, 도 5d에 따른 함수 “get_pk” 를 이용한 맵핑 룰의 선택을 설명할 것이다. 함수 “get_pk” 는 도 3의 알고리즘의 서브알고리즘(312ba)에서의 "pki"의 값을 획득하기 위해 수행될 수 있다는 것을 유념해야 한다. 이에 따라, 도 3의 알고리즘에서 함수 “arith_get_pk”는 함수 “get_pk”로 대체될 수 있다.

[0137] 도 5d에 따른 함수 “get_pk”는 도 17a와 도 17b에 따른 테이블 “ari_s_hash[387]” 및 도 18에 따른 테이블 “ari_gs_hash[225]”을 평가할 수 있다는 것을 또한 유념해야 한다.

[0138] 함수 “get_pk”는, 도 3에 따른 변수 "t" 및 도 3에 따른 변수들 "lev", "lev0"의 조합에 의해 획득될 수 있는 상태 변수 s를 입력 변수로서 수신한다. 함수 “get_pk”는 또한 맵핑 룰 또는 누적 도수 테이블을 지정하는 변수 "pki"의 값을 반환 값으로서 반환하도록 구성된다. 함수 “get_pk”는 상태 값 s를 맵핑 룰 인덱스 값 "pki"로 맵핑하도록 구성된다.

[0139] 함수 “get_pk”는 제1 테이블 평가(540)와, 제2 테이블 평가(544)를 포함한다. 제1 테이블 평가(540)는 참조번호(541)에서 도시된 바와 같이, 변수들 i_min, i_max, 및 i 가 초기화되는 변수 초기화(541)를 포함한다. 제1 테이블 평가(540)는 또한 상태 값 s와 매칭하는 테이블 “ari_s_hash”의 엔트리가 존재하는지 여부에 관한 결정이 행해지는 반복적 테이블 검색(542)을 포함한다. 반복적 테이블 검색(542) 동안에 그러한 매칭이 확인된 경우, 함수 get_pk는 중지되고, 이 함수의 반환 값은 보다 자세하게 설명될 바와 같이, 상태 값 s와 매칭하는 테이블 “ari_s_hash”의 엔트리에 의해 결정된다. 하지만, 반복적 테이블 검색(542)의 과정 동안에 테이블 “ari_s_hash”의 엔트리와 상태 값 s간에 어떠한 완벽한 매칭도 발견되지 않는 경우, 경계 엔트리 체크(543)가 수행된다.

[0140] 이제 제1 테이블 평가(540)의 상세사항으로 관심을 돌리면, 검색 구간은 변수들 i_min 및 i_max에 의해 정의된다는 것을 볼 수 있다. 변수들 i_min 및 i_max에 의해 정의된 구간이 충분히 큰 동안(이것은 조건 i_max-i_min > 1이 충족되는 경우 참일 수 있음)에는 반복적 테이블 검색(542)은 반복된다. 후속하여, 변수 i는, 적어도 대략적으로, 구간의 중간(i=i_min+(i_max-i_min)/2)을 지정하도록 설정된다. 후속하여, 변수 j는, 변수 i에 의해 지정된 어레이 위치에서 어레이 “ari_s_hash”에 의해 결정된 값으로 설정된다(참조번호 542). 여기서 테이블 “ari_s_hash”의 엔트리 각각은 테이블 엔트리와 연관된 상태 값과, 테이블 엔트리와 연관된 맵핑 룰 인덱스 값 모두를 기술한다는 것을 유념해야 한다. 테이블 엔트리와 연관된 상태 값은 테이블 엔트리의 상위 비트들(비트 8~비트 31)에 의해 기술되는 반면에, 맵핑 룰 인덱스 값들은 상기 테이블 엔트리의 하위 비트들(예컨대, 비트 0~비트 7)에 의해 기술된다. 하위 경계 i_min 또는 상위 경계 i_max는 변수 i에 의해 참조된 테이블 “ari_s_hash”의 엔트리 “ari_s_hash[i]”의 최상위 24개 비트들에 의해 기술된 상태 값보다 상태 값 s가 작은지 여부에 의존하여 조정된다. 예를 들어, 상태 값 s가 엔트리 “ari_s_hash[i]”의 최상위 24개 비트들에 의해 기술된 상태 값보다 작은 경우, 테이블 구간의 상위 경계 i_max는 값 i로 설정된다. 따라서, 반복적 테이블 검색(542)의 다음 반복을 위한 테이블 구간은 반복적 테이블 검색(542)의 현재 반복을 위해 이용된 테이블 구간의 아래쪽 절반(i_min에서부터 i_max까지)으로 제한된다. 이와 대조적으로, 상태 값 s가 테이블 엔트리 “ari_s_hash[i]”의 최상위 24개 비트들에 의해 기술된 상태 값들보다 큰 경우, 현재 테이블 구간의 윗쪽 절반(i_min과 i_max사이)이 다음 반복적 테이블 검색을 위한 테이블 구간으로서 이용되도록, 반복적 테이블 검색(542)의 다음 반복을 위한 테이블 구간의 하위 경계 i_min는 값 i로 설정된다. 하지만, 상태 값 s가 테이블 엔트리 “ari_s_hash[i]”의 최상위 24개 비트들에 의해 기술된 상태 값과 동일한 것으로 발견된 경우, 테이블 엔트리 “ari_s_hash[i]”의 최하위 8개 비트들에 의해 기술된 맵핑 룰 인덱스 값은 함수 “get_pk”에 의해 반환되고, 함수는 중지된다.

[0141] 변수들 i_min 및 i_max에 의해 정의된 테이블 구간이 충분히 작을 때 까지 반복적 테이블 검색(542)은 반복된다.

[0142] 경계 엔트리 체크(543)는 (택일적 사항으로서) 반복적 테이블 검색(542)을 보충하도록 실행된다. 반복적 테이블 검색(542)의 완료 이후 인덱스 변수 i가 인덱스 변수 i_max와 동일한 경우, 상태 값 s가 테이블 엔트리 “

ari_s_hash[i_min]" 의 최상위 24개 비트들에 의해 기술된 상태 값과 동일한지 여부에 대한 최종적인 체크가 행해지고, 엔트리 "ari_s_hash[i_min]"의 최하위 8개 비트들에 의해 기술된 맵핑 룰 인덱스 값은, 이 경우에서, 함수 "get_pk" 의 결과로서 반환된다. 이와 대조적으로, 인덱스 변수 i가 인덱스 변수 i_max와 상이한 경우, 상태 값 s가 테이블 엔트리 "ari_s_hash[i_max]" 의 최상위 24개 비트들에 의해 기술된 상태 값과 동일한지 여부에 대한 체크가 행해지고, 상기 테이블 엔트리 "ari_s_hash[i_max]"의 최하위 8개 비트들에 의해 기술된 맵핑 룰 인덱스 값은 이 경우에서 함수 "get_pk" 의 반환 값으로서 반환된다.

[0143] 하지만, 경계 엔트리 체크(543)는 그 전체가 택일적 사항으로서 고려될 수 있다는 것을 유념해야 한다.

[0144] 제1 테이블 평가(540)에 후속하여, 테이블 "ari_s_hash" 의 엔트리들(또는 더 정확하게는 이것의 24개 최상위 비트들)에 의해 기술된 상태 값들 중 하나의 상태 값과 상태 값 s가 동일하게 있는 제1 테이블 평가(540) 동안에 "다이렉트 히트"가 발생하지 않는다면, 제2 테이블 평가(544)가 수행된다.

[0145] 제2 테이블 평가(544)는 참조번호(545)에서 도시된 바와 같이, 인덱스 변수들 i_min, i_max, 및 i 가 초기화되는 변수 초기화(545)를 포함한다. 제2 테이블 평가(544)는 또한 상태 값 s와 동일한 상태 값을 표현하는 엔트리를 찾기 위해 테이블 "ari_gs_hash" 가 검색되는 반복적 테이블 검색(546)을 포함한다. 최종적으로, 제2 테이블 검색(544)은 반환 값 결정(547)을 포함한다.

[0146] 인덱스 변수들 i_min 및 i_max에 의해 정의된 테이블 구간이 충분히 크기만 하다면(예컨대, i_max-i_min > 1 인 경우인 한) 반복적 테이블 검색(546)은 반복된다. 반복적 테이블 검색(546)의 반복에서, 변수 i는 i_min 및 i_max에 의해 정의된 테이블 구간의 중심으로 설정된다(단계 546a). 후속하여, 인덱스 변수 i에 의해 결정된 테이블 위치에서 테이블 "ari_gs_hash"의 엔트리 j가 획득된다(546b). 다시 말하면, 테이블 엔트리 "ari_gs_hash[i]" 는 테이블 인덱스들 i_min 및 i_max에 의해 정의된 현재 테이블 구간의 중심에 있는 테이블 엔트리이다. 후속하여, 반복적 테이블 검색(546)의 다음 반복을 위한 테이블 구간이 결정된다. 이를 위해, 테이블 엔트리 "j=ari_gs_hash[i]" 의 최상위 24개 비트들에 의해 기술된 상태 값보다 상태 값 s가 작은 경우, 테이블 구간의 상위 경계를 기술하는 인덱스 값 i_max는 값 i로 설정된다(546c). 다시 말하면, 반복적 테이블 검색(546)의 다음 반복을 위한 새로운 테이블 구간으로서 현재 테이블 구간의 아래쪽 절반이 선택된다(단계 546c). 그렇지 않고, 테이블 엔트리 "j=ari_gs_hash[i]" 의 최상위 24개 비트들에 의해 기술된 상태 값보다 상태 값 s가 큰 경우, 인덱스 값 i_min은 값 i로 설정된다. 따라서, 반복적 테이블 검색(546)의 다음 반복을 위한 새로운 테이블 구간으로서 현재 테이블 구간의 위쪽 절반이 선택된다(단계 546d). 하지만, 상태 값 s가 테이블 엔트리 "j=ari_gs_hash[i]" 의 최상위 24개 비트들에 의해 기술된 상태 값과 동일한 것으로 발견된 경우, 인덱스 변수 i_max는 변수 i+1로 설정되거나 또는 (i+1이 224보다 큰 경우) 값 224로 설정되고, 반복적 테이블 검색(546)은 중지된다. 하지만, 상태 값 s가 테이블 엔트리 "j=ari_gs_hash[i]" 의 최상위 24개 비트들에 의해 기술된 상태 값과 상이한 경우, 테이블 구간이 너무 작지 않는다면(i_max - i_min ≤ 1), 반복적 테이블 검색(546)은 업데이트된 인덱스 값들 i_min 및 i_max에 의해 정의된 새롭게 설정된 테이블 구간을 갖고 반복된다. 따라서, (i_min 및 i_max에 의해 정의된) 테이블 구간의 구간 사이즈는 "다이렉트 히트"가 검출될 때 까지(s==(j>>8)), 또는 구간이 최소 허용가능한 사이즈에 도달할 때 까지(i_max - i_min ≤ 1) 반복적으로 감소된다. 최종적으로, 반복적 테이블 검색(546)의 중지에 이어서, 테이블 엔트리 "j=ari_gs_hash[i_max]" 가 결정되고, 상기 테이블 엔트리 "j=ari_gs_hash[i_max]" 의 8개 최하위 비트들에 의해 기술된 맵핑 룰 인덱스 값은 함수 "get_pk" 의 반환 값으로서 반환된다. 따라서, 반복적 테이블 검색(546)의 완료 또는 중지 이후 (i_min 및 i_max에 의해 정의된) 테이블 구간의 상위 경계 i_max에 의존하여 맵핑 룰 인덱스 값이 결정된다.

[0147] 반복적 테이블 검색(542, 546)을 모두 이용하는 상술한 테이블 평가들(540, 544)은 주어진 중요 상태의 존재에 대해서 매우 높은 계산 효율성을 갖고 테이블들 "ari_s_hash" 및 "ari_gs_hash"의 검사를 가능하게 해준다. 특히, 테이블 액세스 동작들의 횟수는 최악의 경우에서일지라도 상당히 작게 유지될 수 있다. 테이블 "ari_s_hash" 및 "ari_gs_hash"의 수치적 순서는 적절한 해쉬 값에 대한 검색의 가속화를 가능하게 해준다는 것이 발견되었다. 또한, 테이블들 "ari_s_hash" 및 "ari_gs_hash"에서 탈출 심볼들을 포함하는 것을 필요하지 않으므로 테이블 사이즈는 작게 유지될 수 있다. 따라서, 많은 수의 상이한 상태들이 존재한다 할지라도 효율적인 콘텍스트 해쉬 메커니즘이 구축된다: 제1 스테이지(제1 테이블 평가(540))에서, 다이렉트 히트에 대한 검색이 수행된다(s==(j>>8)).

[0148] 제2 스테이지(제2 테이블 평가(544))에서, 상태 값 s의 범위들은 맵핑 룰 인덱스 값들로 맵핑될 수 있다. 따라서, 테이블 "ari_s_hash"에서 연관된 엔트리가 존재하는 경우의 특별히 중요한 상태들과, 범위 기반 처리가 존재하는 덜 중요한 상태들의 균형잡힌 처리가 수행될 수 있다. 따라서, 함수 "get_pk"는 맵핑 룰 선택의 효

율적인 구현을 구성한다.

[0149] 임의의 보다 상세한 사항에 대해서는, 잘 알려진 프로그래밍 언어 C에 따른 표현으로 함수 “get_pk”의 기능을 표현하는 도 5d의 의사 프로그램 코드를 참조한다.

[0150] 6.5.2 도 5e에 따른 알고리즘을 이용한 맵핑 룰 선택

[0151] 이하에서는, 맵핑 룰의 선택을 위한 또 다른 알고리즘을 도 5e를 참조하여 설명할 것이다. 도 5e에 따른 알고리즘 “arith_get_pk”은, 입력 변수로서, 콘텍스트의 상태를 기술하는 상태 값 s를 수신한다는 것을 유념해야 한다. 함수 “arith_get_pk”는 맵핑 룰(예컨대, 누적 도수 테이블)을 선택하기 위한 인덱스일 수 있는 확률 모델의 인덱스 “pki”를 출력 값 또는 반환 값으로서 제공한다.

[0152] 도 5e에 따른 함수 “arith_get_pk”은 도 3의 함수 “value_decode”의 함수 “arith_get_pk”의 기능을 취할 수 있다는 것을 유념해야 한다.

[0153] 또한, 함수 “arith_get_pk”는, 예컨대 도 20에 따른 테이블 ari_s_hash과, 도 18에 따른 테이블 ari_gs_hash를 평가할 수 있다는 것을 유념해야 한다.

[0154] 도 5e에 따른 함수 “arith_get_pk”는 제1 테이블 평가(550)와 제2 테이블 평가(560)를 포함한다. 제1 테이블 평가(550)에서, 테이블 ari_s_hash의 엔트리 j=ari_s_hash[i]를 획득하기 위해, 상기 테이블에 대한 선형 스캔이 행해진다. 테이블 ari_s_hash의 테이블 엔트리 j=ari_s_hash[i]의 최상위 24개 비트들에 의해 기술된 상태 값이 상태 값 s와 동일한 경우, 상기 확인된 테이블 엔트리 j=ari_s_hash[i]의 최하위 8개 비트들에 의해 기술된 맵핑 룰 인덱스 값 "pki"은 반환되고 함수 “arith_get_pk”은 중지된다. 따라서, "다이렉트 히트"(상태 값 s가 테이블 엔트리 j의 최상위 24개 비트들에 의해 기술된 상태 값과 동일함)가 확인되지 않는다면, 테이블 ari_s_hash의 387개의 모든 엔트리들은 오름차순으로 평가된다.

[0155] 제1 테이블 평가(550)에서 다이렉트 히트가 확인되지 않는 경우, 제2 테이블 평가(560)가 실행된다. 제2 테이블 평가 동안에, 엔트리 인덱스들 i 를 제로로부터 최대값 224까지 선형적으로 증가시키면서 선형 스캔이 수행된다. 제2 테이블 평가 동안에, 테이블 엔트리 j의 24개 최상위 비트들에 의해 표현된 상태 값이 상태 값 s보다 큰지 여부를 결정하도록 테이블 i에 대한 테이블 “ari_gs_hash”의 엔트리 “ari_gs_hash[i]”가 판독되고, 테이블 엔트리 “j=ari_gs_hash[i]”가 평가된다. 이러한 경우라면, 상기 테이블 엔트리 j의 8개 최하위 비트들에 의해 기술된 맵핑 룰 인덱스 값은 함수 “arith_get_pk”의 반환 값으로서 반환되고, 함수 “arith_get_pk”의 실행은 중지된다. 하지만, 상태 값 s가 현재 테이블 엔트리 j=ari_gs_hash[i]의 24개 최상위 비트들에 의해 기술된 상태 값보다 작지 않은 경우, 테이블 ari_gs_hash의 엔트리들에 대한 스캔은 테이블 인덱스 i를 증가시킴으로써 계속된다. 하지만, 상태 값 s가 테이블 ari_gs_hash의 엔트리들에 의해 기술된 상태 값들보다 크거나 또는 이들 중 임의의 상태 값들과 동일한 경우, 테이블 ari_gs_hash의 가장 마지막 엔트리의 8개 최하위 비트들에 의해 정의된 맵핑 룰 인덱스 값 "pki"는 함수 “arith_get_pk”의 반환 값으로서 반환된다.

[0156] 요약하자면, 도 5e에 따른 함수 “arith_get_pk”가 두 단계 해쉬를 수행한다. 제1 단계에서, 다이렉트 히트에 대한 검색이 수행되어, 상태 값 s가 제1 테이블 “ari_s_hash”의 엔트리들 중 임의의 엔트리에 의해 정의된 상태 값과 동일한지 여부를 결정한다. 제1 테이블 평가(550)에서 다이렉트 히트가 확인된 경우, 제1 테이블 “ari_s_hash”로부터 반환 값이 획득되고, 함수 “arith_get_pk”는 중지된다. 하지만, 제1 테이블 평가(550)에서 어떠한 다이렉트 히트도 확인되지 않은 경우, 제2 테이블 평가(560)가 수행된다. 제2 테이블 평가에서, 범위 기반 평가가 수행된다. 제2 테이블 “ari_gs_hash”의 후속하는 엔트리들은 범위들을 정의한다. 하지만, 현재 테이블 엔트리 "j=ari_gs_hash[i]"의 24개 최상위 비트들에 의해 기술된 상태 값이 상태 값 s보다 크다라는 사실에 의해 표시된 범위내에 상태 값 s가 놓여 있다는 것이 발견된 경우, 테이블 엔트리 j=ari_gs_hash[i]의 8개 최하위 비트들에 의해 기술된 맵핑 룰 인덱스 값 "pki"이 반환된다.

[0157] 6.5.3 도 5f에 따른 알고리즘을 이용한 맵핑 룰 선택

[0158] 도 5f에 따른 함수 “get_pk”는 도 5e에 따른 함수 “arith_get_pk”와 실질적으로 등가적이다. 따라서, 위 설명을 참조한다. 보다 세부적인 사항에 대해서는, 도 5f에서의 의사 프로그램 표현을 참조바란다.

[0159] 도 5f에 따른 함수 “get_pk”은 도 3의 함수 “value_decode”에서 “arith_get_pk”이라고 칭해지는 함수를 대신할 수 있다는 것을 유념해야 한다.

[0160] 6.6. 도 5g에 따른 함수 “arith_decode()”

[0161] 이하에서는, 함수 “arith_decode()”의 기능을 도 5g를 참조하여 상세하게 설명할 것이다. 함수 “arith_decode()”는, 시퀀스의 첫번째 심볼인 경우 TRUE를 반환시키고, 그렇지 않은 경우에는 FALSE를 반환시키는 헬퍼 함수“arith_first_symbol (void)”를 이용한다는 것을 유념해야 한다. 함수 “arith_decode()”는 또한 비트스트림의 다음 비트를 얻어서 제공해주는 헬퍼 함수 “arith_get_next_bit(void)”를 이용한다.

[0162] 또한, 함수 “arith_decode()”는 글로벌 변수들 “low”, “high” 및 “value”을 이용한다. 추가로, 함수 “arith_decode()”는 선택된 누적 도수 테이블의 (엘리먼트 인덱스 또는 엔트리 인덱스 0을 갖는) 제1 엔트리 또는 엘리먼트를 향해 가리키는 변수 “cum_freq[]”를 입력 변수로서 수신한다. 또한, 함수 “arith_decode()”는 변수 “cum_freq[]”에 의해 지정된 선택된 누적 도수 테이블의 길이를 표시하는 입력 변수 “cfl”를 이용한다.

[0163] 함수 “arith_decode()”는, 제1 단계로서, 심볼들의 시퀀스 중의 첫번째 심볼이 디코딩중에 있다라고 헬퍼 함수 “arith_first_symbol()”가 표시하는 경우에 수행되는 변수 초기화(570a)를 포함한다. 변수 “value”가 예컨대 복수의 20개 비트들에 의해 표현된 값을 취하도록, 값 초기화 (550a)는 헬퍼 함수 “arith_get_next_bit”를 이용하여 비트스트림으로부터 획득된 상기 20개 비트들에 의존하여 변수 “value”을 초기화한다. 또한, 변수 “low”는 0의 값을 취하도록 초기화되고, 변수 “high”는 1048575의 값을 취하도록 초기화된다.

[0164] 제2 단계(570b)에서, 변수 “range”는 변수들 “high” 및 “low”의 값들간의 차이보다 1만큼 큰 값으로 설정된다. 변수 “cum”는 변수 “low”의 값과 변수 “high”의 값간의 변수 “value”의 값의 상대적인 위치를 표현하는 값으로 설정된다. 따라서, 변수 “cum”는, 예컨대 변수 “value”의 값에 의존하여 0과 2^{16}사이의 값을 취한다.

[0165] 포인터 p는 선택된 누적 도수 테이블의 시작 어드레스보다 1만큼 작은 값으로 초기화된다.

[0166] 알고리즘 “arith_decode()”은 또한 반복적 누적 도수 테이블 검색(570c)을 포함한다. 반복적 누적 도수 테이블 검색은 변수 cfl이 1보다 작거나 또는 이와 동일할 때 까지 반복된다. 반복적 누적 도수 테이블 검색(570c)에서, 포인터 변수 q는 변수 "cfl"의 값의 절반과 포인터 변수 p의 현재 값의 합과 동일한 값으로 설정된다. 엔트리가 포인터 변수 q에 의해 어드레싱되는 선택된 누적 도수 테이블의 엔트리 *q의 값이 변수 “cum”의 값보다 큰 경우, 포인터 변수 p는 포인터 변수 q의 값으로 설정되고, 변수 “cfl”는 증분된다. 최종적으로, 변수 “cfl”은 우측으로 1비트만큼 쉬프트되고, 이로써 변수 “cfl”의 값을 2로 효율적으로 나누고 모듈로 부분은 무시한다.

[0167] 따라서, 확인된 구간내에 값 cum이 놓여 있도록 누적 도수 테이블의 엔트리들에 의해 경계가 정해지는 선택된 누적 도수 테이블내의 구간을 확인하기 위해, 반복적 누적 도수 테이블 검색(570c)은 변수 “cum”의 값을 선택된 누적 도수 테이블의 복수의 엔트리들과 효율적으로 비교한다. 따라서, 선택된 누적 도수 테이블의 엔트리들은 구간들을 정의하며, 각각의 심볼 값은 선택된 누적 도수 테이블의 구간들 각각과 연관된다. 또한, 선택된 누적 도수 테이블 그 전체가 상이한 심볼들(또는 심볼 값들)의 확률 분포를 정의하도록, 누적 도수 테이블의 두 개의 인접한 값들사이의 구간들의 폭들은 상기 구간들과 연관된 심볼들의 확률을 정의한다. 아래에서는 이용가능한 누적 도수 테이블들에 관한 상세사항을 도 19를 참조하여 설명할 것이다.

[0168] 다시 도 5g를 참조하면, 심볼 값은 포인터 변수 p의 값으로부터 유도되며, 심볼 값은 참조번호 570d에서 도시된 바와 같이 유도된다. 따라서, 변수 “symbol”에 의해 표현된 심볼 값을 획득하기 위해 포인터 변수 p의 값과 시작 어드레스 “cum_freq”간의 차이가 평가된다.

[0169] 알고리즘 “arith_decode”은 또한 변수들 “high” 및 “low”의 조정(570e)을 포함한다. 변수 “symbol”에 의해 표현된 심볼 값이 0과 상이한 경우, 참조번호 570e에서 도시된 바와 같이, 변수 “high”가 업데이트된다. 또한, 변수 “low”의 값은 참조번호 570e에서 도시된 바와 같이, 업데이트된다. 변수 “high”는 선택된 누적 도수 테이블의 인덱스 “symbol-1”를 갖는 엔트리, 변수 “range” 및 변수 “low”의 값에 의해 결정된 값으로 설정된다. 변수 “low”는 증가되고, 증가 크기는 인덱스 “symbol”을 갖는 선택된 누적 도수 테이블의 엔트리 및 변수 “range”에 의해 결정된다. 따라서, 변수들 “low”과 “high”의 값들간의 차이는 선택된 누적 도수 테이블의 두 개의 인접한 엔트리들간의 수치적 차이에 의존하여 조정된다.

[0170] 따라서, 낮은 확률을 갖는 심볼 값이 검출된 경우, 변수들 “low”과 “high”의 값들사이의 구간은 좁은 폭으로 감소된다. 이와는 대조적으로, 검출된 심볼 값이 상대적으로 높은 확률을 갖는 경우, 변수들 “low”과 “high”의 값들사이의 구간의 폭은 상대적으로 큰 값으로 설정된다. 다시, 변수들 “low”과 “high”의 값들사

이의 구간의 폭은 검출된 심볼과 누적 도수 테이블의 대응 엔트리들에 의존한다.

[0171] 알고리즘 “arith_decode()”은 또한 단계 570e에서 결정된 구간이 “break” 조건에 도달될 때 까지 반복적으로 쉬프트되고 스케일링되는 구간 재정규화(570f)를 포함한다. 구간 재정규화(570f)에서, 선택적인 하향 쉬프트 동작(570fa)이 수행된다. 변수 “high”가 524286보다 작은 경우, 아무것도 행해지지 않으며, 구간 재정규화는 구간 사이즈 증가 동작(570fb)으로 계속된다. 하지만, 변수 “high”가 524286보다 작지 않고 변수 “low”가 524286 이상인 경우, 변수들 “low” 및 “high”에 의해 정의된 구간이 하향 쉬프트되고, 변수 “value”의 값이 또한 하향 쉬프트되도록, 변수들 “values”, “low” 및 “high”은 모두 524286만큼 감소된다. 하지만, 변수 “high”의 값이 524286보다 작지 않고, 변수 “low”가 524286 이상이 아니며, 변수 “low”가 262143 이상이며, 변수 “high”가 786429보다 작다는 것이 발견된 경우, 변수들 “value”, “low” 및 “high”은 모두 262143만큼 감소되고, 이로써 변수들 “low” 및 “high”의 값들과 또한 변수 “value”의 값사이의 구간을 하향 쉬프트시킨다. 하지만, 위 조건들 중 어느 것도 충족되지 않은 경우, 구간 재정규화는 중지된다.

[0172] 하지만, 단계 570fa에서 평가된 상기 언급된 조건들 중에서 어느 하나라도 충족되는 경우, 구간 증가 동작(570fb)은 실행된다. 구간 증가 동작(570fb)에서, 변수 “low”의 값은 두 배가 된다. 또한, 변수 “high”의 값도 두 배가 되고, 두 배의 결과 1만큼 증가된다. 또한, 변수 “value”의 값도 두 배(좌측으로 1비트만큼 쉬프트됨)가 되고, 헬퍼 함수 “arith_get_next_bit”에 의해 획득된 비트스트림의 비트가 최하위 비트로서 이용된다. 따라서, 변수들 “low”와 “high”의 값사이의 구간의 사이즈는 대략 두 배가 되고, 변수 “value”의 정확도는 비트스트림의 새로운 비트를 이용함으로써 증가된다. 상술한 바와 같이, “break”조건에 도달될 때까지, 즉 변수들 “low”와 “high”의 값들사이의 구간이 충분히 클 때 까지 단계 570fa와 단계 570fb는 반복된다.

[0173] 알고리즘 “arith_decode()”의 기능에 관하여, 변수들 “low”와 “high”의 값들사이의 구간은 단계 570e에서 변수 “cum_freq”에 의해 참조된 누적 도수 테이블의 두 개의 인접한 엔트리들에 의존하여 감소된다는 것을 유념해야 한다. 선택된 누적 도수 테이블의 두 개의 인접한 값들사이의 구간이 작은 경우, 즉 인접한 값들이 비교적 서로 가까운 경우, 단계 570e에서 획득된 변수들 “low”와 “high”의 값들사이의 구간은 비교적 작아질 것이다. 이와 대조적으로, 누적 도수 테이블의 두 개의 인접한 엔트리들이 더욱 이격되는 경우, 단계 570e에서 획득된 변수들 “low”와 “high”의 값들사이의 구간은 비교적 커질 것이다.

[0174] 결과적으로, 단계 570e에서 획득된 변수들 “low”와 “high”의 값들사이의 구간이 비교적 작은 경우, (조건 평가(570fa)의 조건들 중 어떠한 것도 충족되지 않도록) "충분한" 사이즈로 구간을 리스케일링하도록 방대한 갯수의 구간 재정규화 단계들이 실행될 것이다. 따라서, 변수 “value”의 정확도를 증가시키기 위해 비트스트림으로부터 비교적 많은 갯수의 비트들이 이용될 것이다. 이와 대조적으로, 단계 570e에서 획득된 구간 사이즈가 비교적 큰 경우, 변수들 “low” 및 “high” 의 값들 사이의 구간을 "충분한" 크기로 재정규화하기 위해 구간 재정규화 단계들 570fa 및 570fb의 보다 작은 횟수의 반복들만이 필요할 것이다. 따라서, 변수 “value”의 정확도를 증가시키고 다음 심볼의 디코딩을 준비하기 위해 비트스트림으로부터 비교적 작은 갯수의 비트들만이 이용될 것이다.

[0175] 상기의 내용을 요약하자면, 비교적 높은 확률을 포함하고, 선택된 누적 도수 테이블의 엔트리들에 의해 큰 구간이 연관되어진 심볼이 디코딩되는 경우, 후속 심볼의 디코딩을 가능하도록 하기 위해 비교적 적은 갯수의 비트들만이 비트스트림으로부터 판독될 것이다. 이와 대조적으로, 비교적 작은 확률을 포함하고, 선택된 누적 도수 테이블의 엔트리들에 의해 작은 구간이 연관되어진 심볼이 디코딩되는 경우, 다음 심볼의 디코딩을 준비하기 위해 비교적 많은 갯수의 비트들이 비트스트림으로부터 취해질 것이다.

[0176] 따라서, 누적 도수 테이블들의 엔트리들은 상이한 심볼들의 확률들을 반영하고 또한 심볼들의 시퀀스를 디코딩하는데 필요한 비트들의 갯수를 반영한다. 예컨대 상이한 누적 도수 테이블들을 콘텍스트에 의존하여 선택하는 것에 의해 누적 도수 테이블을 콘텍스트에 의존하여, 즉 이전에 디코딩된 심볼들(또는 스펙트럼 값들)에 의존하여 변경시킴으로써, 상이한 심볼들간의 확률적 의존성들이 활용될 수 있고, 이것은 후속(또는 인접한) 심볼들의 특별히 비트레이트 효율적인 인코딩을 가능하게 한다.

[0177] 상기 내용을 요약해보면, (반환 변수 “symbol”에 의해 표현된 심볼 값으로 설정될 수 있는) 최상위 비트플레인 값 m을 결정하기 위해 도 5g를 참조하여 설명한 함수 “arith_decode()”는 함수 “arith_get_pk()”에 의해 반환된 인덱스 “pki”에 대응하는 누적 도수 테이블 “arith_cf_m[pki][]”로 호출된다.

[0178] 6.7 탈출 메커니즘

[0179] 함수 “arith_decode ()”에 의해 심볼 값으로서 반환된 디코딩된 최상위 비트플레인 값 m이 탈출 심볼 “ARITH_ESCAPE”인 경우에는, 추가적인 최상위 비트플레인 값 m이 디코딩되고 변수 “lev”은 1만큼 증분된다. 따라서, 디코딩될 하위 비트플레인들의 갯수뿐만이 아니라 최상위 비트플레인 값 m의 수치적 중요도(numeric significance)에 관한 정보가 획득된다.

[0180] 탈출 심볼 “ARITH_ESCAPE”이 디코딩되는 경우, 레벨 변수 “lev”는 1만큼 증가된다. 따라서, 최상위 비트들(비트 24 및 그 위)에 의해 표현된 값이 알고리즘 312ba의 다음 반복을 위해 증가되도록 함수 “arith_get_pk”로 입력되는 상태 값이 또한 수정된다.

[0181] 6.8. 도 5h에 따른 콘텍스트 업데이트

[0182] 스펙트럼 값이 완전히 디코딩되면, 즉 최하위 비트플레인들 모두가 추가되면, 콘텍스트 테이블 q 및 qs는 함수 “arith_update_context(a,i,lg)”를 호출함으로써 업데이트된다. 이하에서는, 함수 “arith_update_context(a,i,lg)”에 관한 상세사항을 상기 함수의 의사 프로그램 코드 표현을 도시하는 도 5h를 참조하여 설명할 것이다.

[0183] 함수 “arith_update_context()”는 입력 변수들로서, 디코딩되고 양자화된 스펙트럼 계수 a, 디코딩될 스펙트럼 값(또는 디코딩된 스펙트럼 값)의 인덱스 i, 및 현재 오디오 프레임과 연관된 스펙트럼 값들(또는 계수들)의 갯수 lg를 수신한다.

[0184] 단계 580에서, 현재 디코딩되고 양자화된 스펙트럼 값(또는 계수) a는 콘텍스트 테이블 또는 콘텍스트 어레이 q 내로 복사된다. 따라서, 콘텍스트 테이블 q의 엔트리 q[1][i]는 a로 설정된다. 또한, 변수 “a0”는 “a”의 값으로 설정된다.

[0185] 단계 582에서, 콘텍스트 테이블 q의 레벨 값 q[1][i].l이 결정된다. 디폴트에 의해, 콘텍스트 테이블 q의 레벨 값 q[1][i].l은 제로로 설정된다. 하지만, 현재 코딩된 스펙트럼 값 a의 절대 값이 4보다 큰 경우, 레벨 값 q[1][i].l은 증분된다. 각각의 증분으로, 변수 “a”는 1비트만큼 우측으로 쉬프트된다. 레벨 값 q[1][i].l 의 증분은 변수 a0의 절대 값이 4보다 작거나 또는 이와 동일할 때 까지 반복된다.

[0186] 단계 584에서, 콘텍스트 테이블 q의 2비트 콘텍스트 값 q[1][i].c이 설정된다. 현재 디코딩된 스펙트럼 값 a가 제로와 동일한 경우 2비트 콘텍스트 값 q[1][i].c는 제로의 값으로 설정된다. 그렇지 않고, 디코딩된 스펙트럼 값 a의 절대 값이 1보다 작거나, 또는 1과 동일한 경우, 2비트 콘텍스트 값 q[1][i].c은 1로 설정된다. 그렇지 않고, 현재 디코딩된 스펙트럼 값 a의 절대 값이 3보다 작거나, 또는 3과 동일한 경우, 2비트 콘텍스트 값 q[1][i].c은 2로 설정된다. 그렇지 않은 경우, 즉 현재 디코딩된 스펙트럼 값 a의 절대 값이 3보다 큰 경우, 2비트 콘텍스트 값 q[1][i].c은 3로 설정된다. 따라서, 현재 디코딩된 스펙트럼 계수 a의 매우 거친 양자화에 의해 2비트 콘텍스트 값 q[1][i].c가 획득된다.

[0187] 현재 디코딩된 스펙트럼 값의 인덱스 i가 프레임에서의 계수들(스펙트럼 값들)의 갯수 lg와 동일한 경우, 즉 프레임의 가장 마지막 스펙트럼 값이 디코딩되고 코어 모드가 선형 예측 영역 코어 모드(이것은 “core_mode==1”에 의해 표시됨)인 경우에만 수행되는 후속 단계 586에서, 엔트리들 q[1][j].c은 콘텍스트 테이블 qs[k]내로 복사된다. 현재 프레임에서의 스펙트럼 값들의 갯수 lg가 엔트리들 q[1][j].c을 콘텍스트 테이블 qs[k]로 복사하기 위해 고려되도록, 참조번호 586에서 도시된 바와 같이 복사가 수행된다. 또한, 변수 “previous_lg”는 값 1024를 취한다.

[0188] 하지만, 대안적으로, 현재 디코딩된 스펙트럼 계수의 인덱스 i가 lg의 값에 도달하고 코어 모드가 주파수 영역 코어 모드(이것은 “core_mode==0”에 의해 표시됨)인 경우 콘텍스트 테이블 q의 엔트리들 q[1][j].c은 콘텍스트 테이블 qs[j]내로 복사된다.

[0189] 이 경우, 변수 “previous_lg”는 프레임에서의 스펙트럼 값들의 갯수 lg와 1024의 값 사이에서 최소값으로 설정된다.

[0190] 6.9 디코딩 처리의 요약

[0191] 이후에는, 디코딩 처리를 간략하게 요약할 것이다. 세부사항에 대해서는 위 설명 및 또한 도 3, 도 4 및 도 5a 내지 도 5i를 참조한다.

[0192] 양자화된 스펙트럼 계수들 a 는 최저 주파수 계수로부터 시작해서 최고 주파수 계수로 진행하면서 무잡음방식으로 코딩되어 전달된다.

[0193] 진보된 오디오 코딩(advanced-audio coding; AAC)으로부터의 계수들은 어레이 “x_ac_quant[g][win][sfb][bin]”에 저장되고, 무잡음 코딩 코드워드들이 수신되어 어레이에 저장된 순서로 디코딩될 때, bin 이 가장 급속하게 증분하는 인덱스이고 g가 가장 느리게 증분하는 인덱스가 되도록 무잡음 코딩 코드워드들의 전달 순서가 정해진다. 인덱스 bin은 주파수 빈을 지정한다. 인덱스 “sfb”는 스케일 인자 대역들을 지정한다. 인덱스 “win”는 윈도우들을 지정한다. 인덱스 “g”는 오디오 프레임들을 지정한다.

[0194] 변환 코딩된 여기로부터의 계수들은 어레이 x_tcx_invquant[win][bin]에 직접 저장되고, 무잡음 코딩 코드워드들이 수신되어 어레이에 저장된 순서로 디코딩될 때, “bin” 이 가장 급속하게 증분하는 인덱스이고 “win” 이 가장 느리게 증분하는 인덱스가 되도록 무잡음 코딩 코드워드의 전달 순서가 정해진다.

[0195] 먼저, 콘텍스트 테이블 또는 어레이 “qs”에 저장된 과거 콘텍스트와 (콘텍스트 테이블 또는 어레이 q에 저장된) 현재 프레임 q의 콘텍스트간의 맵핑이 행해진다. 과거 콘텍스트 “qs”는 주파수 라인 당 (또는 주파수 빈 당) 2비트로 저장된다.

[0196] 콘텍스트 테이블 “qs”에 저장된 과거 콘텍스트와 콘텍스트 테이블 "q"에 저장된 현재 프레임의 콘텍스트간의 맵핑이 함수 “arith_map_context()”(이것의 의사 프로그램 코드 표현은 도 5a에서 도시됨)을 이용하여 수행된다.

[0197] 무잡음 디코더는 서명된 양자화 스펙트럼 계수들 "a"을 출력한다.

[0198] 첫번째로, 콘텍스트의 상태가 디코딩할 양자화된 스펙트럼 계수들을 둘러싸는 이전에 디코딩된 스펙트럼 계수들에 기초하여 계산된다. 콘텍스트 s의 상태는 함수 “arith_get_context()”에 의해 반환된 값의 처음 24개 비트들에 대응한다. 반환된 값의 24번째 비트를 넘어서는 비트들은 예측된 비트 플레인 레벨 lev0에 대응한다. 변수 "lev"은 lev0으로 초기화된다. 함수 “arith_get_context”의 의사 프로그램 코드 표현이 도 5b와 도 5c에서 도시된다.

[0199] 상태 s와 예측된 레벨 "lev"이 알려지면, 최상위 2비트 와이즈 플레인 m은, 콘텍스트 상태에 대응하는 확률 모델에 대응한 적절한 누적 도수 테이블이 제공되는 함수 “arith_decode()”를 이용하여 디코딩된다.

[0200] 대응은 함수 “arith_get_pk()”에 의해 행해진다.

[0201] 함수 “arith_get_pk()”의 의사 프로그램 코드 표현이 도 5e에서 도시된다.

[0202] 함수 “arith_get_pk()”를 대신할 수 있는 또 다른 함수 “get_pk”의 의사 프로그램 코드가 도 5f에서 도시된다. 함수 “arith_get_pk()”를 대신할 수 있는 또 다른 함수 “get_pk”의 의사 프로그램 코드가 도 5d에서 도시된다.

[0203] 값 m은 누적 도수 테이블 "arith_cf_m[pki][]"로 호출된 함수 “arith_decode()”를 이용하여 디코딩되며, 여기서 "pki"는 함수 “arith_get_pk()” (또는, 대안적으로 함수 “get_pk()”)에 의해 반환된 인덱스에 대응한다.

[0204] 산술 코더는 스케일링을 갖춘 태크 생성 방법을 이용한 정수 구현이다(예컨대, K. Sayood의 “Introduction to Data Compression”(제3판, 2006, Elsevier Inc)을 참조하라). 도 5g에서 도시된 의사 C 코드는 이용된 알고리즘을 기술한다.

[0205] 디코딩된 값 m이 탈출 심볼 “ARITH_ESCAPE”인 경우, 또 다른 값 m이 디코딩되고 변수 “lev”은 1만큼 증분된다. 값 m이 탈출 심볼 “ARITH_ESCAPE”이 아닌 경우, 누적 도수 테이블 “arith_cf_r[]”로 함수 “arith_decode()”을 "lev"회 호출함으로써 나머지 비트플레인들이 최상위 레벨에서부터 최하위 레벨까지 디코딩된다. 상기 누적 도수 테이블 "arith_cf_r[]"은 예컨대 균일한 확률 분포를 기술할 수 있다.

[0206] 디코딩된 비트플레인들 r은 다음의 방법으로 이전에 디코딩된 값 m의 개량을 허용한다:

[0207] a = m;

[0208] for (i=0; i<lev;i++) {

[0209] r = arith_decode (arith_cf_r,2);

[0210] a = (a<<1)|(r&1);

[0211] }

[0212] 스펙트럼 양자화된 계수 a가 완전히 디코딩되면, 콘텍스트 테이블들 q, 또는 저장된 콘텍스트 qs는 디코딩될 다음의 양자화된 스펙트럼 계수들을 위해 함수 “arith_update_context()”에 의해 업데이트된다.

[0213] 함수 “arith_update_context()”의 의사 프로그램 코드 표현이 도 5h에서 도시된다.

[0214] 또한, 정의들의 범례가 도 5i에서 도시된다.

[0215] 7. 맵핑 테이블들

[0216] 본 발명에 따른 실시예에서, 특별히 유리한 테이블들 “ari_s_hash”, “ari_gs_hash” 및 “ari_cf_m”은 도 5d를 참조하여 설명된 함수 “get_pk”의 실행을 위해 이용되거나, 또는 도 5e를 참조하여 설명된 함수 “arith_get_pk”의 실행을 위해 이용되거나, 또는 도 5f를 참조하여 설명된 함수 “get_pk”의 실행을 위해 이용되거나, 또는 도 5g를 참조하여 설명하였던 함수 “arith_decode”의 실행을 위해 이용된다.

[0217] 7.1. 도 17에 따른 테이블 “ari_s_hash[387]”

[0218] 도 5d를 참조하여 설명하였던 함수 “get_pk”에 의해 이용된 테이블 “ari_s_hash”의 특히 유리한 구현의 내용이 도 17의 테이블에서 도시된다. 도 17의 테이블은 테이블 “ari_s_hash[387]”의 387개 엔트리들을 나열한 것임을 유념해야 한다. 도 17의 테이블 표현은, 첫번째 값 “0x00000200”이 엘리먼트 인덱스(또는 테이블 인덱스) 0을 갖는 테이블 엔트리 “ari_s_hash[0]”에 대응하고, 마지막 값 “0x03D0713D”이 엘리먼트 인덱스 또는 테이블 인덱스 386를 갖는 테이블 엔트리 “ari_s_hash[386]”에 대응하도록 하는 엘리먼트 인덱스들의 순서로 엘리먼트들을 도시한다는 것을 또한 유념해야 한다. 여기서 테이블 “ari_s_hash”의 테이블 엔트리들이 16진법 형식으로 표현된다라고 “0x”가 표시한다는 것을 또한 유념해야 한다. 더군다나, 함수 “get_pk”의 제1 테이블 평가(540)의 실행을 허용하기 위해 도 17에 따른 테이블 “ari_s_hash”의 테이블 엔트리들은 수치적 순서로 배열된다.

[0219] 테이블 “ari_s_hash”의 테이블 엔트리들의 최상위 24개 비트들은 상태 값들을 표현하는 반면에, 최하위 8개 비트들은 맵핑 룰 인덱스 값들 pki를 표현한다는 것을 또한 유념해야 한다.

[0220] 따라서, 테이블 “ari_s_hash”의 엔트리들은 맵핑 룰 인덱스 값 “pki”으로의 상태 값의 "다이렉트 히트" 맵핑을 기술한다.

[0221] 7.2. 도 18에 따른 테이블 “ari_gs_hash”

[0222] 테이블 “ari_gs_hash”의 특히 유리한 실시예의 내용이 도 18의 테이블에서 도시된다. 여기서 도 18의 테이블은 테이블 “ari_gs_hash”의 엔트리들을 나열한 것임을 유념해야 한다. 상기 엔트리들은 예컨대 "i"로 지정된 1차원의 정수형태 엔트리 인덱스(이것은 또한 "엘리먼트 인덱스" 또는 "어레이 인덱스" 또는 "테이블 인덱스"로서 지정됨)에 의해 참조표시된다. 총 225개 엔트리들을 포함한 테이블 “ari_gs_hash”는 도 5d에서 설명된 함수 “get_pk”의 제2 테이블 평가(544)에 의해 이용하기에 적합하다는 것을 유념해야 한다.

[0223] 테이블 “ari_gs_hash”의 엔트리들은 제로와 224사이에서 테이블 인덱스 값들 i에 대하여 테이블 인덱스 i의 오름차순으로 나열된다는 것을 유념해야 한다. 용어 “0x”는 테이블 엔트리들이 16진법 형식으로 기술된다는 것을 표시한다. 따라서, 첫번째 테이블 엔트리 “0X00000401”는 테이블 인덱스 0을 갖는 테이블 엔트리 “ari_gs_hash[0]”에 대응하며, 가장 마지막 테이블 엔트리 “0Xffffff3f”는 테이블 인덱스 224를 갖는 테이블 엔트리 “ari_gs_hash[224]”에 대응한다.

[0224] 테이블 엔트리들이 함수 “get_pk”의 제2 테이블 평가(544)에 적합하도록 테이블 엔트리들은 수치적 오름차순으로 순서화된다는 것을 또한 유념해야 한다. 테이블 “ari_gs_hash”의 테이블 엔트리들의 최상위 24개 비트들은 상태 값들의 범위들사이의 경계들을 기술하고, 테이블 엔트리들의 8개의 최하위 비트들은 24개 최상위 비트들에 의해 정의된 상태 값들의 범위들과 연관된 맵핑 룰 인덱스 값들 “pki”을 기술한다.

[0225] 7.3. 도 19에 따른 테이블 “ari_cf_m”

[0226] 도 19는 64개의 누적 도수 테이블들 “ari_cf_m[pki][9]”의 세트를 도시하며, 이 테이블들 중에서 하나의 테이블은 예컨대 함수 “arith_decode”의 실행을 위해, 즉 최상위 비트플레인 값의 디코딩을 위해, 오디오 인코더(100, 700) 또는 오디오 디코더(200, 800)에 의해 선택된다. 도 19에서 도시된 64개 누적 도수 테이블들 중 선택된 테이블은 함수 “arith_decode()”의 실행시 테이블 “cum_freq[]”의 함수를 취한다.

[0227] 도 19로부터 살펴볼 수 있는 바와 같이, 각각의 라인은 9개의 엔트리들을 갖는 누적 도수 테이블을 표현한다.

예를 들어, 첫번째 라인(1910)은 “pki=0”에 대한 누적 도수 테이블의 9개 엔트리들을 표현한다. 두번째 라인(1912)은 “pki=1”에 대한 누적 도수 테이블의 9개 엔트리들을 표현한다. 마지막으로, 64번째 라인(1964)은 “pki=63”에 대한 누적 도수 테이블의 9개 엔트리들을 표현한다. 따라서, 도 19는 “pki=0”에서 “pki=63”까지에 대한 64개의 상이한 누적 도수 테이블들을 효과적으로 표현하며, 64개 누적 도수 테이블들 각각은 단일 라인에 의해 표현되고 상기 누적 도수 테이블들 각각은 9개의 엔트리들을 갖는다.

[0228] 라인(예컨대, 라인(1910) 또는 라인(1912) 또는 라인(1964)) 내에서, 가장왼쪽의 값은 누적 도수 테이블의 첫번째 엔트리를 기술하고, 가장오른쪽의 값은 누적 도수 테이블의 마지막 엔트리를 기술한다.

[0229] 따라서, 도 19의 테이블 표현의 각각의 라인(1910, 1912, 1964)은 도 5g에 따른 함수 “arith_decode”에 의한 이용을 위한 누적 도수 테이블의 엔트리들을 표현한다. 함수 “arith_decode”의 입력 변수 “cum_freq[]”는 테이블 “ari_cf_m”의 (9개 엔트리들의 개별적인 라인들에 의해 표현된) 64개 누적 도수 테이블들 중 어느 것이 현재 스펙트럼 계수들의 디코딩을 위해 이용되어야 하는지를 기술한다.

[0230] 7.4. 도 20에 따른 테이블 “ari_s_hash”

[0231] 도 20은 도 5e 또는 도 5f에 따른 대안적인 함수 “arith_get_pk()” 또는 “get_pk()”와 함께 이용될 수 있는 테이블 “ari_s_hash”에 대한 대안을 도시한다.

[0232] 도 20에 따른 테이블 “ari_s_hash”은 386개의 엔트리들을 포함하며, 이 엔트리들은 도 20에서 테이블 인덱스의 오름차순으로 나열된다. 따라서, 첫번째 테이블 값 “0x0090D52E”은 테이블 인덱스 0을 갖는 테이블 엔트리 “ari_s_hash[0]”에 대응하며, 가장 마지막 테이블 엔트리 “0x03D0513C”는 테이블 인덱스 386을 갖는 테이블 엔트리 “ari_s_hash[386]”에 대응한다.

[0233] 용어 “0x”는 테이블 엔트리들이 16진법 형식으로 표현된다는 것을 표시한다. 테이블 “ari_s_hash”의 엔트리들 중의 24개 최상위 비트들은 중요 상태들을 기술하며, 테이블 “ari_s_hash”의 엔트리들 중의 8개의 최하위 비트들은 맵핑 룰 인덱스 값들을 기술한다.

[0234] 따라서, 테이블 “ari_s_hash”의 엔트리들은 맵핑 룰 인덱스 값들 “pki”으로의 중요 상태들의 맵핑을 기술한다.

[0235] 8. 성능 평가 및 장점들

[0236] 본 발명에 따른 실시예들은 계산 복잡성, 메모리 요건 및 코딩 효율성간의 향상된 트레이드오프를 획득하기 위해 상술한 바와 같이, 업데이트된 함수들(또는 알고리즘들) 및 업데이트된 테이블들의 세트를 이용한다.

[0237] 일반적으로, 본 발명에 따른 실시예들은 향상된 스펙트럼 무잡음 코딩을 생성한다.

[0238] 본 설명은 스펙트럼 계수들의 향상된 스펙트럼 무잡음 코딩에 관한 CE를 위한 실시예들을 기술한다. 제안된 방식은 USAC 드래프트 표준의 작업 드래프트 4에서 기술된 바와 같은 "오리지널" 콘텍스트 기반 산술 코딩에 기초하지만, 무잡음 코딩 성능을 유지하면서 메모리 요건(RAM, ROM)을 상당히 감소시킨다. WD3(즉, USAC 드래프트 표준의 작업 드래프트 3에 따라 비트스트림을 제공하는 오디오 인코더의 출력)의 무손실 트랜스코딩이 가능한 것으로 판명되었다. 여기서 설명된 방식은, 일반적으로, 메모리 요건과 인코딩 성능간의 추가적인 대안적 트레이트오프를 허용하는 확장성을 갖는다. 본 발명에 따른 실시예들은 USAC 드래프트 표준의 작업 드래프트 4에서 이용된 스펙트럼 무잡음 코딩 방식을 대체시키는 것을 목적으로 한다.

[0239] 여기서 설명된 산술 코딩 방식은 USAC 드래프트 표준의 작업 드래프트 4(WD4) 또는 기준 모델 0(RM0)에서의 코딩 방식에 기초한다. 주파수 또는 시간상에서의 이전의 스펙트럼 계수들은 콘텍스트를 모델링한다. 이 콘텍스트는 산술 코더(인코더 또는 디코더)를 위한 누적 도수 테이블들의 선택을 위해 이용된다. WD4에 따른 실시예와 비교하여, 콘텍스트 모델링은 한층 향상된 것이고 심볼 확률들을 유지하는 테이블들이 리트레이닝되었다. 상이한 확률 모델들의 갯수는 32개에서 64개로 증가되었다.

[0240] 본 발명에 따른 실시예들은 테이블 사이즈들(데이터 ROM 수요량)을 32비트 또는 3600 바이트 길이의 900 워드까지 감소시킨다. 이와 대조적으로, USAC 드래프트 표준의 WD4에 따른 실시예들은 16894.5 워드 또는 76578 바이트를 필요로 한다. 본 발명에 따른 몇몇 실시예들에서는, 정적 RAM 수요량이 코어 코더 채널 당 666 워드(2664 바이트)로부터 72 워드(288 바이트)로 감소된다. 이와 동시에, 이것은 코딩 성능을 완전히 보존하며, 심지어 9개의 모든 동작점들에 대한 총 데이터레이트와 비교하여 대략 1.04% 내지 1.39%의 이득에 도달할 수 있다. 모든 작업 드래프트 3(WD3) 비트스트림들은 비트 저장소 제약에 영향을 주는 것 없이 무손실 방식으로 트랜스코딩될

수 있다.

[0241] 본 발명의 실시예들에 따른 제안된 방식은 확장성이 있어서, 메모리 수요량과 코딩 성능간의 유연한 트레이드오프가 가능하다. 테이블 사이즈를 증가시킴으로써 코딩 이득은 한층 더 증가될 수 있다.

[0242] 이하에서는, 여기서 설명된 개념의 장점들을 보다 잘 이해할 수 있게 하기 위해 USAC 드래프트 표준의 WD4에 따른 코딩 개념의 간략한 설명을 제공할 것이다. USAC WD4에서는, 양자화된 스펙트럼 계수들의 무잡음 코딩을 위해 콘텍스트 기반 산술 코딩 방식이 이용된다. 콘텍스트로서, 디코딩된 스펙트럼 계수들이 이용되는데, 이것은 주파수 및 시간상에서 이전의 것이다. WD4에 따르면, 최대 16개의 스펙트럼 계수들이 콘텍스트로서 이용되며, 이 중 12개는 시간상 이전의 것이다. 콘텍스트를 위해 이용되고 디코딩될 스펙트럼 계수들 모두는 4개 튜플로서 그룹화된다(즉, 주파수에서 이웃하는 네 개의 스펙트럼 계수들, 도 10a 참조). 콘텍스트는 감소되고 누적 도수 테이블에 맵핑되며, 그런 후 스펙트럼 계수들의 다음 4개 튜플을 디코딩하기 위해 이용된다.

[0243] 완전한 WD4 무잡음 코딩 방식의 경우, 16894.5 워드(67578 바이트)의 메모리 수요량(ROM)이 요구된다. 추가적으로, 다음 프레임을 위한 상태들을 저장하기 위해 코어 코더 채널 당 정적 ROM의 666 워드(2664 바이트)가 필요하다.

[0244] 도 11a의 테이블 표현은 USAC WD4 산술 코딩 방식에서 이용된 테이블들을 기술한다.

[0245] 완전한 USAC WD4 디코더의 총 메모리 수요량은 프로그램 코드가 없는 데이터 ROM에 대해 37000 워드(148000 바이트)가 되고 정적 RAM에 대해서는 10000 내지 17000 워드가 되는 것으로 추정된다. 무잡음 코더 테이블들은 총 데이터 ROM 수요량의 대략 45%를 소모시킨다는 것을 명확히 살펴볼 수 있다. 가장 큰 개별 테이블은 이미 4096 워드(16384 바이트)를 소모한다.

[0246] 모든 테이블들의 조합 및 커다란 개별적인 테이블들의 사이즈는 모두 저가의 포터블 디바이스들(예컨대, ARM9e, TIC64xx 등)을 위한 고정포인트 칩들에 의해 제공되는 전형적인 캐시 사이즈(8~32 kByte의 일반적인 범위에 놓여 있음)를 초과한다는 것이 발견되어 왔다. 이것은 테이블들의 세트가 아마도 데이터에 대한 빠른 랜덤 액세스를 가능하게 해주는 고속 데이터 RAM에 저장될 수 없다는 것을 의미한다. 이것은 전체적인 디코딩 처리를 감속시킨다.

[0247] 이하에서는, 제안된 새로운 방식을 간략하게 설명할 것이다.

[0248] 상기에서 언급한 문제들을 극복하기 위해, USAC 드래프트 표준의 WD4에서의 코딩 방식을 대체하는 향상된 무잡음 코딩 방식이 제안된다. 이것은, 콘텍스트 기반 산술 코딩 방식으로서, USAC 드래프트 표준의 WD4 방식에 기초하지만, 콘텍스트로부터의 누적 도수 테이블들의 유도를 위한 수정된 방식에 그 특징을 둔다. 더 나아가, 콘텍스트 유도 및 심볼 코딩은 (USAC 드래프트 표준의 WD4에서의 4개 튜플과는 대조적으로) 단일 스펙트럼 계수의 입도(granularity)로 수행된다. 전체적으로, (적어도 몇몇의 경우들에서) 7개의 스펙트럼 계수들이 콘텍스트를 위해 이용된다. 맵핑에서의 감축에 의해, 총 64개(WD4에서는: 32개) 확률 모델들 또는 누적 도수 테이블들 중 하나가 선택된다.

[0249] 도 10b는 제안된 방식에서 이용되는, 상태 계산을 위한 콘텍스트의 그래픽 표현을 도시한다(제로 영역 검출을 위해 이용된 콘텍스트는 도 10b에서는 도시되지 않는다).

[0250] 이하에서는, 제안된 코딩 방식을 이용함으로써 달성될 수 있는 메모리 수요량의 감축에 관한 간략한 설명을 제공할 것이다. 제안된 새로운 방식은 900 워드(3600 바이트)의 총 ROM 수요량을 나타낸다(제안된 코딩 방식에서 이용된 테이블들을 기술하는 도 11b의 테이블을 참조바란다).

[0251] USAC 드래프트 표준의 WD4에서의 무잡음 코딩 방식의 ROM 수요량과 비교하여, ROM 수요량은 15994.5 워드(64978 바이트)만큼 감소된다(USAC 드래프트 표준의 WD4에서의 무잡음 코딩 방식 및 제안된 무잡음 코딩 방식의 ROM 수요량의 그래픽 표현을 도시한 도 12a를 참조바란다). 이것은 완전한 USAC 디코더의 총체적인 ROM 수요량을 대략 37000 워드에서 대략 21000 워드로 감소시키거나, 또는 43% 이상 감소시킨다(본 제안구성 뿐만이 아니라, USAC 드래프트 표준의 WD4에 따른 총체적인 USAC 디코더 데이터 ROM 수요량의 그래픽 표현을 도시한 도 12b를 참조바란다).

[0252] 더 나아가, 다음 프레임에서 콘텍스트 유도를 위해 필요한 정보의 양(정적 RAM)이 또한 감소된다. WD4에 따르면, 분해능 10비트의 4튜플 당 그룹 인덱스에 추가되는 일반적인 16비트의 분해능을 갖는 계수들의 완전한 세트(최대 1152개)가 저장될 필요가 있으며, 이것은 코어 코더 채널(완전한 USAC WD4 디코더: 대략 10000 내지

17000 워드) 당 666 워드(2664 바이트)까지 합산된다.

[0253] 본 발명에 따른 실시예들에서 이용된 새로운 방식은 영구적 정보를 스펙트럼 계수 당 단지 2비트(이것은 코어 코더 채널 당 전체적으로 72워드(288바이트)까지 합산된다)로 감소시킨다. 정적 메모리에 대한 수요량은 594워드(2376 바이트)만큼 감소될 수 있다.

[0254] 이하에서는, 코딩 효율성의 잠재적 증가에 관한 몇가지 세부사항들을 설명할 것이다. 새로운 제안구성에 따른 실시예들의 코딩 효율성은 USAC 드래프트 표준의 WD3에 따른 참조 퀄리티 비트스트림들과 대조되었다. 이러한 비교는 참조 소프트웨어 디코더에 기초하여 트랜스코더에 의해 수행되었다. 제안된 코딩 방식과 USAC 드래프트 표준의 WD3에 따른 무잡음 코딩의 비교에 관한 세부사항에 대해서는, 테스트 장치의 개략적 표현을 도시한 도 9를 참조한다.

[0255] USAC 드래프트 표준의 WD3 또는 WD4에 따른 실시예들과 비교하여 본 발명에 따른 실시예들에서는 메모리 수요량이 대폭적으로 감소되며, 코딩 효율성은 유지될 뿐만이 아니라 약간 증가된다. 코딩 효율성은 평균적으로 1.04% 내지 1.39%만큼 증가된다. 세부사항에 대해서는, 본 발명의 실시예에 따른 오디오 코더(예컨대, USAC 오디오 코더)와 작업 드래프트 산술 코더를 이용한 USAC 코더에 의해 생성된 평균 비트레이트의 테이블 표현을 도시한 도 13a의 테이블을 참조한다.

[0256] 비트 저장소 충진 레벨의 측정에 의하면, 제안된 무잡음 코딩은 매 동작점마다 WD3 비트스트림을 무잡음방식으로 트랜스코딩할 수 있다는 것을 보여주었다. 세부사항에 대해서는, 본 발명의 실시예에 따른 오디오 코더와 USAC WD3에 따른 오디오 코더를 위한 비트 저장소 제어의 테이블 표현을 도시한 도 13b의 테이블을 참조한다.

[0257] 동작 모드 당 평균 비트레이트, 프레임 단위의 최소, 최대, 및 평균 비트레이트, 및 프레임 단위의 최상/최악의 성능에 관한 상세사항이 도 14, 도 15 및 도 16의 테이블들에서 발견될 수 있으며, 도 14의 테이블은 본 발명의 실시예에 따른 오디오 코더와 USAC WD3에 따른 오디오 코더를 위한 평균 비트레이트의 테이블 표현을 도시하며, 도 15의 테이블은 USAC 오디오 코더의 프레임 단위의 최소, 최대 및 평균 비트레이트들의 테이블 표현을 도시하며, 도 16의 테이블은 프레임 단위의 최상의 경우 및 최악의 경우의 테이블 표현을 도시한다.

[0258] 또한, 본 발명에 따른 실시예들은 우수한 확장성을 제공한다는 것을 유념해야 한다. 테이블 사이즈를 조정함으로써, 메모리 요건, 계산적 복잡성 및 코딩 효율성간의 트레이드오프가 요건들에 따라 조정될 수 있다.

[0259] 9. 비트스트림 구문(Syntax)

[0260] 9.1. 스펙트럼 무잡음 코더의 페이로드

[0261] 이하에서는, 스펙트럼 무잡음 코더의 페이로드에 관한 몇가지 세부사항들을 설명할 것이다. 몇몇 실시예들에서, 예컨대 소위 말하는 "선형 예측 영역" 코딩 모드 및 "주파수 영역" 코딩 모드와 같은 복수의 상이한 코딩 모드들이 존재한다. 선형 예측 영역 코딩 모드에서, 오디오 신호의 선형 예측 분석에 기초하여 노이즈 셰이핑이 수행되고, 노이즈 셰이핑된 신호는 주파수 영역에서 인코딩된다. 주파수 영역 모드에서, 노이즈 셰이핑은 심리음향 분석에 기초하여 수행되고, 노이즈 셰이핑된 버전의 오디오 콘텐츠는 주파수 영역에서 인코딩된다.

[0262] "선형 예측 영역" 코딩된 신호와 "주파수 영역" 코딩된 신호 모두로부터의 스펙트럼 계수들은 스칼라 양자화되고 그런 후 조정된 콘텍스트 의존적 산술 코딩에 의해 무잡음 코딩된다. 양자화된 계수들은 최저 주파수에서부터 최고 주파수까지 전달된다. 각각의 개별적인 양자화된 계수들은 최상위 2비트 와이즈 플레인 m과, 나머지 하위 비트 플레인들 r로 분할된다. 값 m은 계수의 근접성에 따라 코딩된다. 나머지 하위 비트 플레인들 r은 콘텍스트를 고려하지 않고서 엔트로피 인코딩된다. 값 m과 값 r은 산술 코더의 심볼들을 형성한다.

[0263] 상세한 산술 디코딩 프로시저를 여기서 설명한다.

[0264] 9.2. 구문 엘리먼트들

[0265] 이하에서는, 산술적으로 인코딩된 스펙트럼 정보를 운송하는 비트스트림의 비트스트림 구문을 도 6a 내지 도 6h를 참조하여 설명할 것이다.

[0266] 도 6a는 소위 말하는 USAC 미가공 데이터 블록(“usac_raw_data_block()”)의 구문 표현을 도시한다.

[0267] USAC 미가공 데이터 블록은 하나 이상의 단일 채널 엘리먼트들(“single_channel_element()”) 및/또는 하나 이상의 채널 쌍 엘리먼트들(“channel_pair_element()”)을 포함한다.

[0268] 이제 도 6b를 참조하여, 단일 채널 엘리먼트의 구문을 설명한다. 단일 채널 엘리먼트는 코어 모드에 의존하여

선형 예측 영역 채널 스트림(“lpd_channel_stream ()”) 또는 주파수 영역 채널 스트림(“fd_channel_stream ()”)을 포함한다.

[0269] 도 6c는 채널 쌍 엘리먼트의 구문 표현을 도시한다. 채널 쌍 엘리먼트는 코어 모드 정보(“core_mode0”, “core_mode1”)를 포함한다. 또한, 채널 쌍 엘리먼트는 구성 정보 “ics_info()”를 포함할 수 있다. 추가적으로, 코어 모드 정보에 의존하여, 채널 쌍 엘리먼트는 채널들 중의 제1 채널과 연관된 주파수 영역 채널 스트림 또는 선형 예측 영역 채널 스트림을 포함하고, 채널 쌍 엘리먼트는 또한 채널들 중의 제2 채널과 연관된 주파수 영역 채널 스트림 또는 선형 예측 영역 채널 스트림을 포함한다.

[0270] 도 6d에 구문 표현이 도시되어 있는 구성 정보 “ics_info()”는 복수의 상이한 구성 정보 아이템들을 포함하는데, 이것은 본 발명과 특별히 관련성이 있지 않다.

[0271] 도 6e에 구문 표현이 도시되어 있는 주파수 영역 채널 스트림(“fd_channel_stream ()”)은 이득 정보(“global_gain”) 및 구성 정보(“ics_info ()”)를 포함한다. 또한, 주파수 영역 채널 스트림은 상이한 스케일 인자 대역들의 스펙트럼 값들의 스케일링을 위해 이용된 스케일 인자들을 기술하는 스케일 인자 데이터(“scale_factor_data ()”)를 포함하며, 이 스케일 인자 데이터는 예컨대 스케일러(150) 및 리스케일러(240)에 의해 적용된다. 주파수 영역 채널 스트림은 또한 산술적으로 인코딩된 스펙트럼 값들을 표현한 산술적으로 코딩된 스펙트럼 데이터(“ac_spectral_data ()”)를 포함한다.

[0272] 도 6f에 구문 표현이 도시되어 있는 산술적으로 코딩된 스펙트럼 데이터(“ac_spectral_data()”)는 상술한 바와 같은, 콘텍스트를 선택적으로 재설정하는데 이용되는 택일적인 산술 재설정 플래그(“arith_reset_flag”)를 포함한다. 또한, 산술적으로 코딩된 스펙트럼 데이터는 산술적으로 코딩된 스펙트럼 값들을 운송하는 복수의 산술 데이터 블록들(“arith_data”)을 포함한다. 산술적으로 코딩된 데이터 블록들의 구조는 주파수 대역들의 갯수(이것은 변수 “num_bands”로 표현됨) 및 또한 산술 재설정 플래그의 상태(이후에 설명할 것임)에 의존한다.

[0273] 산술적으로 코딩된 데이터 블록들의 구문 표현을 도시한 도 6g를 참조하여 산술적으로 인코딩된 데이터 블록의 구조를 설명할 것이다. 산술적으로 코딩된 데이터 블록 내의 데이터 표현은 인코딩될 스펙트럼 값들의 갯수 lg, 산술 재설정 플래그의 상태, 및 콘텍스트, 즉 이전에 인코딩된 스펙트럼 값들에 의존한다.

[0274] 현재의 스펙트럼 값들의 세트의 인코딩을 위한 콘텍스트는 참조번호 660에서 도시된 콘텍스트 결정 알고리즘에 따라 결정된다. 콘텍스트 결정 알고리즘과 관련한 상세사항은 도 5a를 참조하여 상술하였다. 산술적으로 인코딩된 데이터 블록은 코드워드들의 lg개 세트들을 포함하며, 이 코드워드들의 세트 각각은 스펙트럼 값을 표현한다. 코드워드들의 세트는 1개와 20개 사이의 비트들을 이용하여 스펙트럼 값의 최상위 비트플레인 값 m을 표현한 산술 코드워드 “acod_m [pki][m]”를 포함한다. 또한, 정확한 표현을 위해 스펙트럼 값이 최상위 비트플레인보다 많은 비트 플레인들을 필요한 경우 코드워드들의 세트는 하나 이상의 코드워드들 “acod_r[r]”을 포함한다. 코드워드 “acod_r[r]”는 1개와 20개 사이의 비트들을 이용하여 하위 비트플레인을 표현한다.

[0275] 하지만, 스펙트럼 값의 적절한 표현을 위해 (최상위 비트플레인에 더하여) 하나 이상의 하위 비트플레인들이 필요한 경우, 이것은 하나 이상의 산술 탈출 코드워드들(“ARITH_ESCAPE”)을 이용하여 시그널링된다. 따라서, 스펙트럼 값에 대하여, 얼마나 많은 비트플레인들(최상위 비트플레인 및 잠재적으로는 하나 이상의 추가적인 하위 비트플레인들)이 필요한지가 결정되는 것으로 일반적으로 말해질 수 있다. 하나 이상의 하위 비트플레인들이 필요한 경우, 이것은 하나 이상의 산술 탈출 코드워드들 “acod_m [pki][ARITH_ESCAPE]”에 의해 시그널링되며, 이 코드워드들은 현재 선택된 누적 도수 테이블, 변수 pki에 의해 주어진 누적 도수 테이블 인덱스에 따라 인코딩된다. 또한, 하나 이상의 산술 탈출 코드워드들이 비트스트림내에 포함된 경우, 참조번호들 664, 662에서 살펴볼 수 있는 바와 같이, 콘텍스트가 조정된다. 하나 이상의 산술 탈출 코드워드들에 이어서, 참조번호 663에서 도시된 산술 코드워드 “acod_m [pki][m]”가 비트스트림내에 포함되며, 여기서 pki는 (산술 탈출 코드워드들의 포함에 의해 유발된 콘텍스트 조정을 고려한) 현재 유효한 확률 모델 인덱스를 가리키며, m은 인코딩되거나 또는 디코딩될 스펙트럼 값의 최상위 비트플레인 값을 가리킨다.

[0276] 상술한 바와 같이, 임의의 하위 비트플레인들의 존재는 하나 이상의 코드워드들 “acod_r[r]”의 존재를 초래하며, 이 코드워드들 각각은 1비트의 최하위 비트플레인을 표현한다. 하나 이상의 코드워드들 “acod_r[r]”은 일정하면서 콘텍스트 독립적인 대응 누적 도수 테이블에 따라 인코딩된다.

[0277] 또한, 두 개의 후속하는 스펙트럼 값들의 인코딩을 위한 콘텍스트가 일반적으로 상이하도록, 참조번호 668에서 도시된 바와 같이, 콘텍스트는 각각의 스펙트럼 값의 인코딩 이후에 업데이트된다는 것을 유념해야 한다.

[0278] 도 6h는 정의들의 범례를 도시하며, 산술적으로 인코딩된 데이터 블록의 구문을 정의하는 엘리먼트들을 도와준

다.

[0279] 상기를 요약하자면, 오디오 코더(100)에 의해 제공될 수 있고, 오디오 디코더(200)에 의해 평가될 수 있는 비트스트림 포맷이 설명되었다. 산술적으로 인코딩된 스펙트럼 값들의 비트스트림은 상술한 디코딩 알고리즘에 들어맞도록 인코딩된다.

[0280] 또한, 인코딩은 디코딩의 역 동작이라는 것을 일반적으로 유념해야 하며, 이에 따라 인코더는 상술한 테이블들을 이용하여 테이블 검색을 수행하되, 이러한 테이블 검색은 디코더에 의해 수행된 테이블 검색에 대해 대략적으로 정반대라는 것이 일반적으로 추정될 수 있다. 일반적으로, 디코딩 알고리즘 및/또는 희망하는 비트스트림 구문을 알고 있는 본 발명분야의 당업자는 산술 디코더에 의해 요구되고 비트스트림 구문에서 정의된 데이터를 제공하는 산술 인코더를 손쉽게 설계할 수 있을 것이라고 말할 수 있다.

[0281] 10. 도 21 및 도 22에 따른 추가적인 실시예들

[0282] 이하에서는, 본 발명에 따른 몇가지 추가적인 단순화된 실시예들을 설명할 것이다.

[0283] 도 21은 본 발명의 실시예에 따른 오디오 인코더(2100)의 개략적인 블록도를 도시한다. 오디오 인코더(2100)는 입력 오디오 정보(2110)를 수신하고, 이를 기초로, 인코딩된 오디오 정보(2112)를 제공하도록 구성된다. 오디오 인코더(2100)는 에너지 압축 시간 영역-대-주파수 영역 컨버터를 포함하며, 이 에너지 압축 시간 영역-대-주파수 영역 컨버터는, 주파수 영역 오디오 표현이 스펙트럼 값들의 세트(예컨대, 스펙트럼 값들 a)를 포함하도록, 입력 오디오 표현(2110)의 시간 영역 표현(2122)을 수신하고, 이에 기초하여, 주파수 영역 오디오 표현(2124)을 제공하도록 구성된다. 오디오 신호 인코더(2100)는 또한 가변 길이 코드워드를 이용하여 스펙트럼 값들(2124) 또는 이것의 사전처리된 버전을 인코딩하도록 구성된 산술 인코더(2130)를 포함한다. 산술 인코더(2130)는 스펙트럼 값, 또는 스펙트럼 값의 최상위 비트플레인의 값을 코드 값(예컨대, 가변 길이 코드워드를 표현한 코드 값)에 맵핑하도록 구성된다.

[0284] 산술 인코더는 맵핑 룰 선택(2132) 및 콘텍스트 값 결정(2136)을 포함한다. 산술 인코더는 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값(2134)에 의존하여 (가변 길이 코드워드를 표현할 수 있는)코드 값으로의 스펙트럼 값(2124), 또는 스펙트럼 값(2124)의 최상위 비트플레인의 맵핑을 기술하는 맵핑 룰을 선택하도록 구성된다. 산술 디코더는 이전에 인코딩된 복수의 스펙트럼 값들에 의존하여, 맵핑 룰 선택(2132)을 위해 이용되는, 수치적 현재 콘텍스트 값(2134)을 결정하도록 구성된다. 산술 인코더, 또는 보다 정확하게는 맵핑 룰 선택(2132)은, 반복 구간 사이즈 감소를 이용하여 적어도 하나의 테이블을 평가하고, 선택된 맵핑 룰을 기술하는 맵핑 룰 인덱스 값(2133)을 유도하기 위해, 수치적 현재 콘텍스트 값(2134)이 테이블의 엔트리에 의해 기술된 테이블 콘텍스트 값과 동일하거나 또는 테이블의 엔트리들에 의해 기술된 구간 내에 놓여있는지 여부를 결정하도록 구성된다. 따라서, 맵핑(2131)은 수치적 현재 콘텍스트 값(2134)에 의존하여 높은 계산 효율성을 갖고 선택될 수 있다.

[0285] 도 22는 본 발명의 또 다른 실시예에 따른 오디오 신호 디코더(2200)의 개략적인 블록도를 도시한다. 오디오 신호 디코더(2200)는 인코딩된 오디오 정보(2210)를 수신하고, 이를 기초로, 디코딩된 오디오 정보(2212)를 제공하도록 구성된다. 오디오 신호 디코더(2200)는 스펙트럼 값들의 산술적으로 인코딩된 표현(2222)을 수신하고, 이를 기초로, 복수의 디코딩된 스펙트럼 값들(2224)(예컨대, 디코딩된 스펙트럼 값들 a)을 제공하도록 구성된 산술 디코더(2220)를 포함한다. 오디오 신호 디코더(2200)는 또한 디코딩된 스펙트럼 값들(2224)을 수신하고, 디코딩된 오디오 정보(2212)를 획득하기 위해, 디코딩된 스펙트럼 값들을 이용하여 시간 영역 오디오 표현을 제공하도록 구성된 주파수 영역-대-시간 영역 컨버터(2230)를 포함한다.

[0286] 산술 디코더(2220)는 코드 값(예컨대, 인코딩된 오디오 정보를 표현하는 비트스트림으로부터 추출된 코드 값)을 심볼 코드(이것은 예컨대 디코딩된 스펙트럼 값, 또는 디코딩된 스펙트럼 값의 최상위 비트플레인을 기술할 수 있음)로 맵핑하는데 이용되는 맵핑(2225)을 포함한다. 산술 디코더는 맵핑 룰 선택 정보(2227)를 맵핑(2225)에 제공하는 맵핑 룰 선택(2226)을 더 포함한다. 산술 디코더(2220)는 또한 수치적 현재 콘텍스트 값(2229)을 맵핑 룰 선택(2226)에 제공하는 콘텍스트 값 결정(2228)을 포함한다.

[0287] 산술 디코더(2220)는 콘텍스트 상태에 의존하여 코드 값(예컨대, 인코딩된 오디오 정보를 표현하는 비트스트림으로부터 추출된 코드 값)의 심볼 코드(예컨대, 디코딩된 스펙트럼 값을 표현한 수치 값, 또는 디코딩된 스펙트럼 값의 최상위 비트플레인을 표현한 수치 값)로의 맵핑을 기술하는 맵핑 룰을 선택하도록 구성된다. 산술 디코더는 이전에 디코딩된 복수의 스펙트럼 값들에 의존하여 현재 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값을 결정하도록 구성된다. 또한, 산술 디코더 (또는 보다 정확하게는 맵핑 룰 선택(2226))은, 반복 구간 사이

즈 감소를 이용하여 적어도 하나의 테이블을 평가하고, 선택된 맵핑 룰을 기술하는 맵핑 룰 인덱스 값(2227)을 유도하기 위해, 수치적 현재 콘텍스트 값(2229)이 테이블의 엔트리에 의해 기술된 테이블 콘텍스트 값과 동일하거나 또는 테이블의 엔트리들에 의해 기술된 구간 내에 놓여있는지 여부를 결정하도록 구성된다. 따라서, 맵핑(2225)에서 적용된 맵핑 룰은 계산 효율적인 방식으로 선택될 수 있다.

[0288] 11. 구현 대안책들

[0289] 비록 몇몇 양태들은 장치의 관점에서 설명되었지만, 이러한 양태들은 또한 대응 방법의 설명을 나타낸다는 것이 명백하며, 여기서 블록 또는 디바이스는 방법 단계 또는 방법 단계의 특징에 대응한다. 마찬가지로, 방법 단계의 관점에서 설명된 양태들은 또한 대응하는 장치의 대응하는 블록 또는 아이템 또는 특징의 설명을 나타낸다. 방법 단계들 모두 또는 그 일부는 예컨대, 마이크로프로세서, 프로그램가능 컴퓨터 또는 전자 회로와 같은 하드웨어 장치에 의해(또는 이것을 이용하여) 실행될 수 있다. 몇몇 실시예들에서, 가장 중요한 방법 단계들 중의 몇몇의 하나 이상의 방법 단계들은 이러한 장치에 의해 실행될 수 있다.

[0290] 본 발명의 인코딩된 오디오 신호는 디지털 저장 매체상에 저장될 수 있거나 또는 인터넷과 같은 무선 전송 매체 또는 유선 전송 매체와 같은 전송 매체를 통해 전송될 수 있다.

[0291] 일정한 구현 요건에 따라, 본 발명의 실시예들은 하드웨어나 소프트웨어로 구현될 수 있다. 이러한 구현은 전자적으로 판독가능한 제어 신호들이 저장되어 있으며, 각각의 방법이 수행되도록 프로그램가능한 컴퓨터 시스템과 협동하는(또는 이와 협동가능한) 디지털 저장 매체, 예컨대 플로피 디스크, DVD, 블루레이, CD, ROM, PROM, EPROM, EEPROM 또는 FLASH 메모리를 이용하여 수행될 수 있다. 그러므로, 디지털 저장 매체는 컴퓨터로 판독가능할 수 있다.

[0292] 본 발명에 따른 몇몇의 실시예들은 여기서 설명된 방법들 중 하나의 방법이 수행되도록, 프로그램가능한 컴퓨터 시스템과 협동할 수 있는 전자적으로 판독가능한 제어 신호들을 갖는 데이터 캐리어를 포함한다.

[0293] 일반적으로, 본 발명의 실시예들은 컴퓨터 프로그램 제품이 컴퓨터 상에서 구동될 때 본 방법들 중 하나의 방법을 수행하기 위해 동작되는 프로그램 코드를 갖는 컴퓨터 프로그램 제품으로서 구현될 수 있다. 프로그램 코드는 예컨대 머신 판독가능한 캐리어 상에 저장될 수 있다.

[0294] 다른 실시예들은 머신 판독가능한 캐리어 상에서 저장되는, 여기서 설명된 방법들 중 하나의 방법을 수행하기 위한 컴퓨터 프로그램을 포함한다.

[0295] 다시 말하면, 본 발명의 방법의 실시예는, 따라서, 컴퓨터 상에서 컴퓨터 프로그램이 구동될 때, 여기서 설명된 방법들 중 하나의 방법을 수행하기 위한 프로그램 코드를 갖는 컴퓨터 프로그램이다.

[0296] 본 발명의 방법들의 추가적인 실시예는, 이에 따라 여기서 설명된 방법들 중 하나의 방법을 수행하기 위한 컴퓨터 프로그램이 기록되어 있는 데이터 캐리어(또는 디지털 저장 매체, 또는 컴퓨터 판독가능한 매체)이다.

[0297] 본 발명의 방법의 추가적인 실시예는, 이에 따라 여기서 설명된 방법들 중 하나의 방법을 수행하기 위한 컴퓨터 프로그램을 표현한 신호들의 시퀀스 또는 데이터 스트림이다. 신호들의 시퀀스 또는 데이터 스트림은 데이터 통신 접속, 예컨대 인터넷을 통해 전송되도록 구성될 수 있다.

[0298] 추가적인 실시예는 여기서 설명된 방법들 중 하나의 방법을 수행하도록 구성되거나 조정된 프로세싱 수단, 예컨대 컴퓨터, 또는 프로그램가능 논리 디바이스를 포함한다.

[0299] 추가적인 실시예는 여기서 설명된 방법들 중 하나의 방법을 수행하기 위한 컴퓨터 프로그램이 설치된 컴퓨터를 포함한다.

[0300] 몇몇의 실시예들에서, 프로그램가능한 논리 디바이스(예컨대 필드 프로그램가능한 게이트 어레이)는 여기서 설명된 방법들의 기능들 모두 또는 그 일부를 수행하기 위해 이용될 수 있다. 몇몇의 실시예들에서, 여기서 설명된 방법들 중 하나의 방법을 수행하기 위해 필드 프로그램가능한 게이트 어레이가 마이크로프로세서와 협동할 수 있다. 일반적으로, 본 방법들은 바람직하게는 임의의 하드웨어 장치에 의해 수행된다.

[0301] 상술한 실시예들은 본 발명의 원리들에 대한 일례에 불과하다. 여기서 설명된 구성 및 상세사항의 수정 및 변형은 본 발명분야의 당업자에게 자명할 것으로 이해된다. 그러므로, 본 발명은 계류중인 본 특허 청구항들의 범위에 의해서만 제한이 되며 여기서의 실시예들의 설명 및 해설을 통해 제시된 특정한 세부사항들에 의해서는 제한되지 않는다는 것이 본 취지이다.

[0302] 상기 내용을 상기 특별한 실시예들을 참조하여 특별하게 도시하고 설명하였지만, 본 발명의 사상과 범위으로부터 벗어나지 않고서 형태적으로 및 세부적으로 다양한 다른 변경들이 취해질 수 있다는 것이 본 발명분야의 당업자에게는 이해될 것이다. 이러한 다양한 변경들은 여기서 개시된 광범위한 개념으로부터 벗어나지 않고서 상이한 실시예들에 적응하도록 행해질 수 있고 아래의 청구항들에 의해 이해될 수 있다는 것이 이해될 것이다.

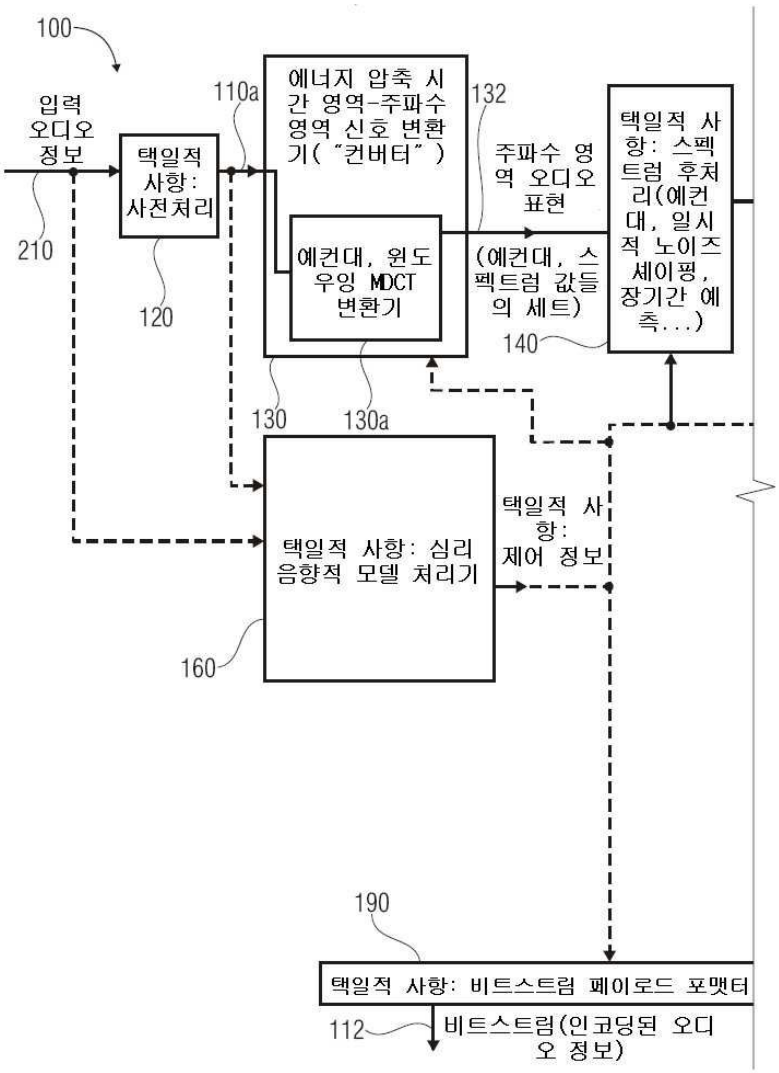
[0303] 12. 결론

[0304] 결론적으로, 본 발명에 따른 실시예들은 향상된 스펙트럼 무잡음 코딩 방식을 생성한다는 것을 알 수 있다. 새로운 제안에 따른 실시예들은 16894.5 워드에서 900 워드로(ROM), 그리고 666 워드에서 72 워드로(코어 코더 채널 당 정적 RAM) 메모리 수요량의 상당한 감소를 가능하게 해준다. 이것은 일 실시예에서 완전한 시스템의 데이터 ROM 수요량의 대략 43%만큼의 감소를 가능하게 해준다. 이와 동시에, 코딩 성능은 완전히 유지될 뿐만이 아니라, 심지어 평균적으로 증가된다. WD3(또는, USAC 드래프트 표준의 WD3에 따라 제공된 비트스트림)의 무손실 트랜스코딩이 가능한 것으로 판명되었다. 따라서, 본 발명에 따른 실시예는 여기서 설명된 무잡음 디코딩을 USAC 드래프트 표준의 장래에 다가올 작업 드래프트에서 채용함으로써 획득된다.

[0305] 요약해보면, 실시예에서 제안된 새로운 무잡음 코딩은 도 6g에서 도시된 비트스트림 엘리먼트의 구문 “arith_data()”과 관련하여, 도 5h에서 도시되고 상술된 스펙트럼 무잡음 코더의 페이로드와 관련하여, 상술한 스펙트럼 무잡음 코딩과 관련하여, 도 4에서 도시된 상태 계산을 위한 콘텍스트와 관련하여, 도 5i에서 도시된 정의들과 관련하여, 도 5a, 도 5b, 도 5c, 도 5e, 도 5g, 도 5h와 관련하여 상술한 디코딩 처리와 관련하여, 도 17, 도 18, 도 20에서 도시된 테이블들과 관련하여, 및 도 5d에서 도시된 함수 “get_pk”와 관련하여 MPEG USAC 작업 드래프트에서의 수정을 불러일으킬 수 있다. 하지만, 대안적으로, 도 20에 따른 테이블 “ari_s_hash”은 도 17의 테이블 “ari_s_hash”을 대신하여 이용될 수 있으며, 도 5f의 함수 “get_pk”는 도 5d에 따른 함수 “get_pk”를 대신하여 이용될 수 있다.

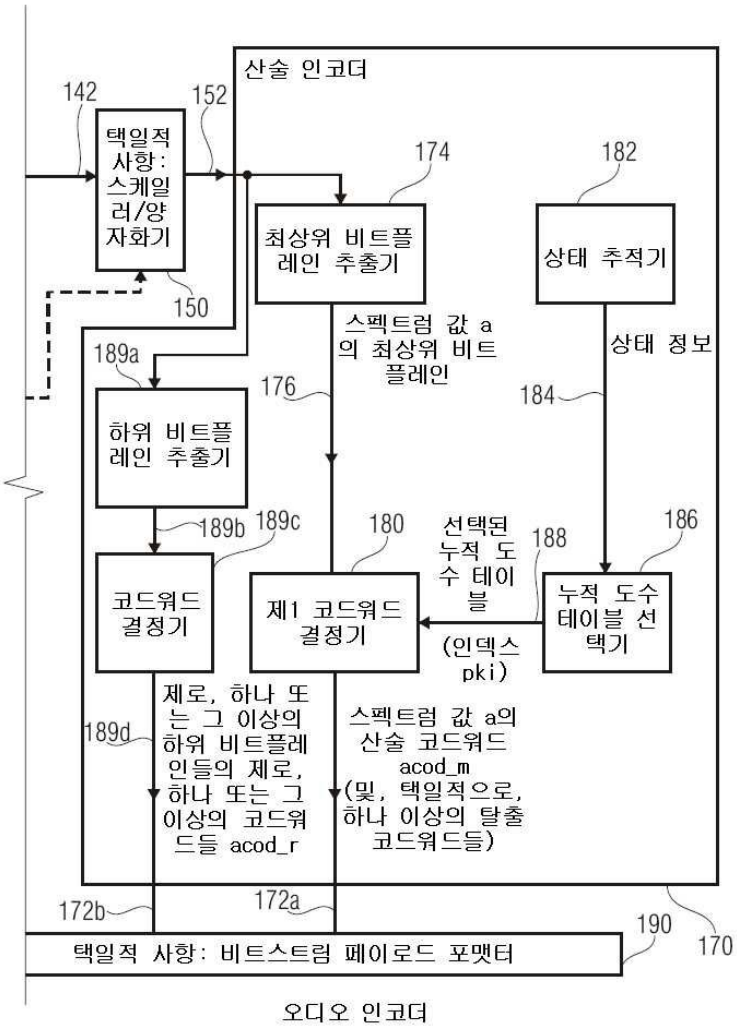
도면

도면1a

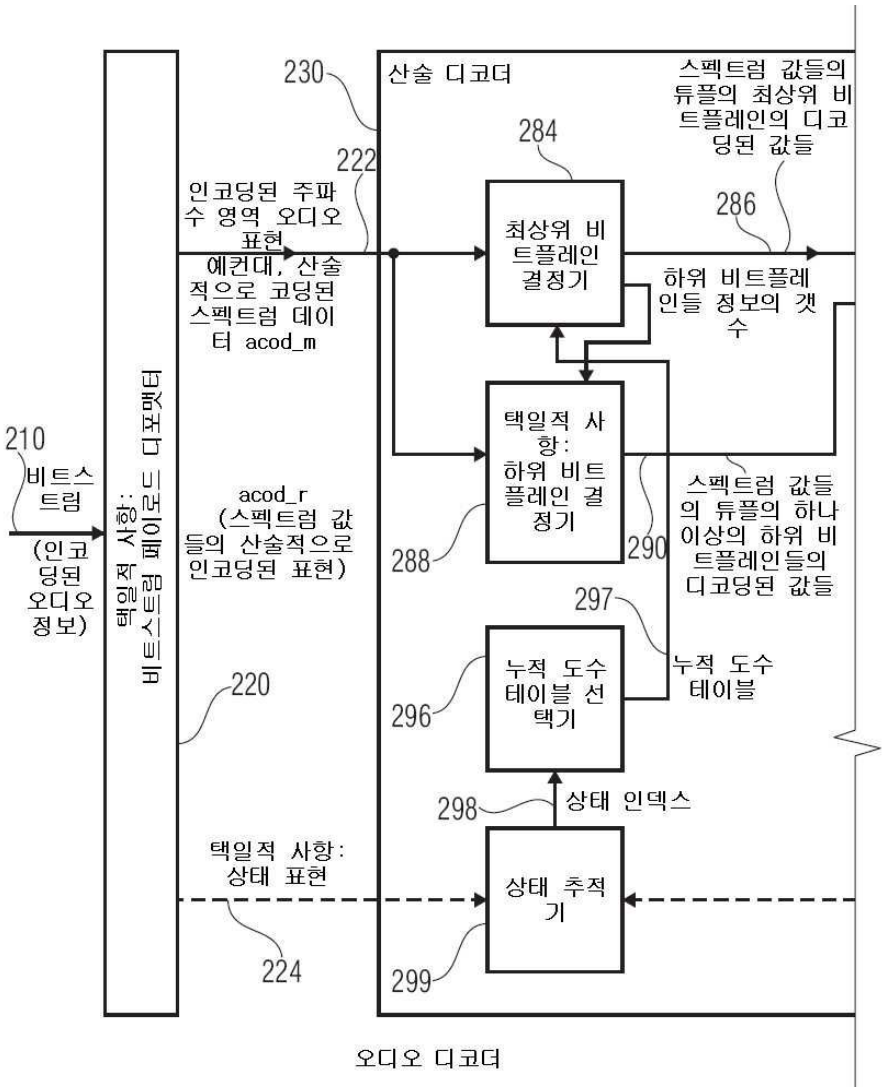


오디오 인코더

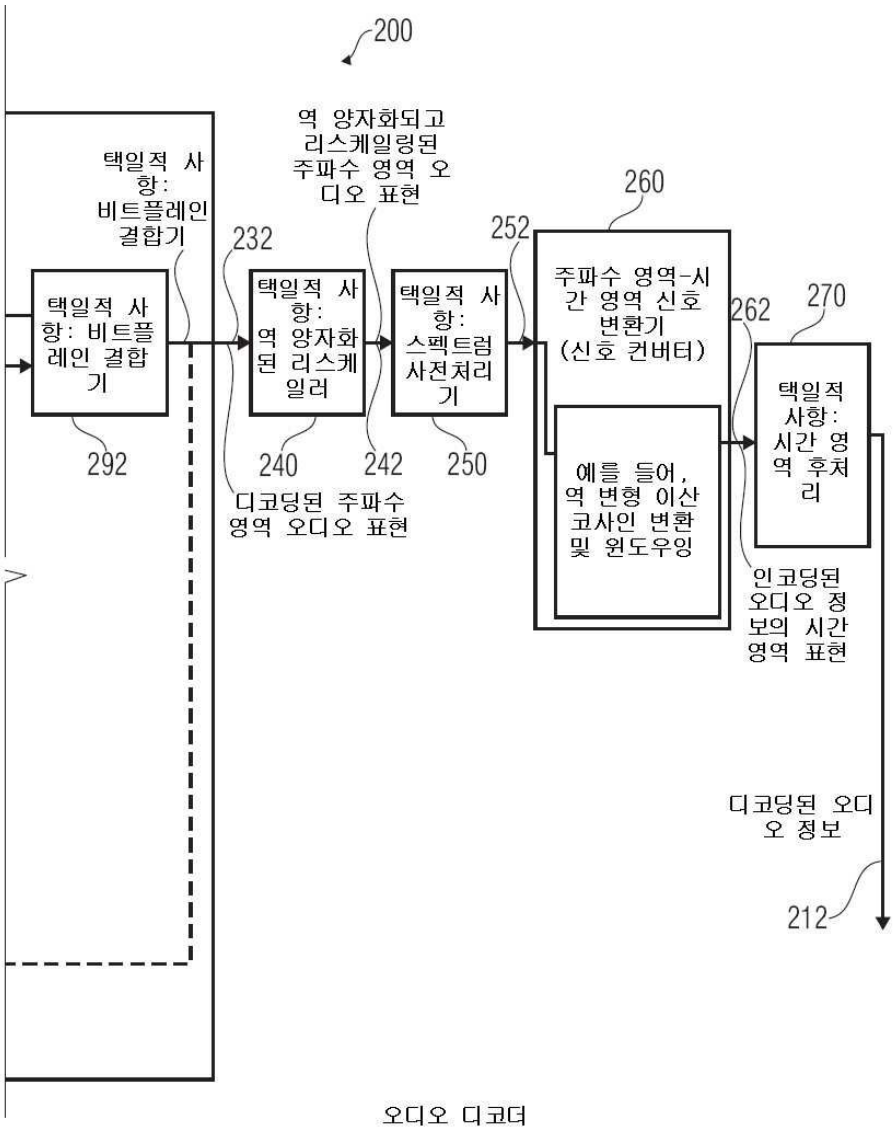
도면1b


산술 인코더
142
152
택일적 사항: 스케일러/양자화기
150
174
최상위 비트플레인 추출기
182
상태 추적기
스펙트럼 값 a의 최상위 비트플레인
상태 정보
189a
176
184
하위 비트플레인 추출기
189b
189c
180
선택된 누적 도수 테이블
188
186
코드워드 결정기
제1 코드워드 결정기
누적 도수 테이블 선택기
(인덱스 pki)
제로, 하나 또는 그 이상의 하위 비트플레인들의 제로, 하나 또는 그 이상의 코드워드들 acod_r
189d
스펙트럼 값 a의 산술 코드워드 acod_m (및, 택일적으로, 하나 이상의 탈출 코드워드들)
172b
172a
190
170
택일적 사항: 비트스트림 페이로드 포맷터
오디오 인코더

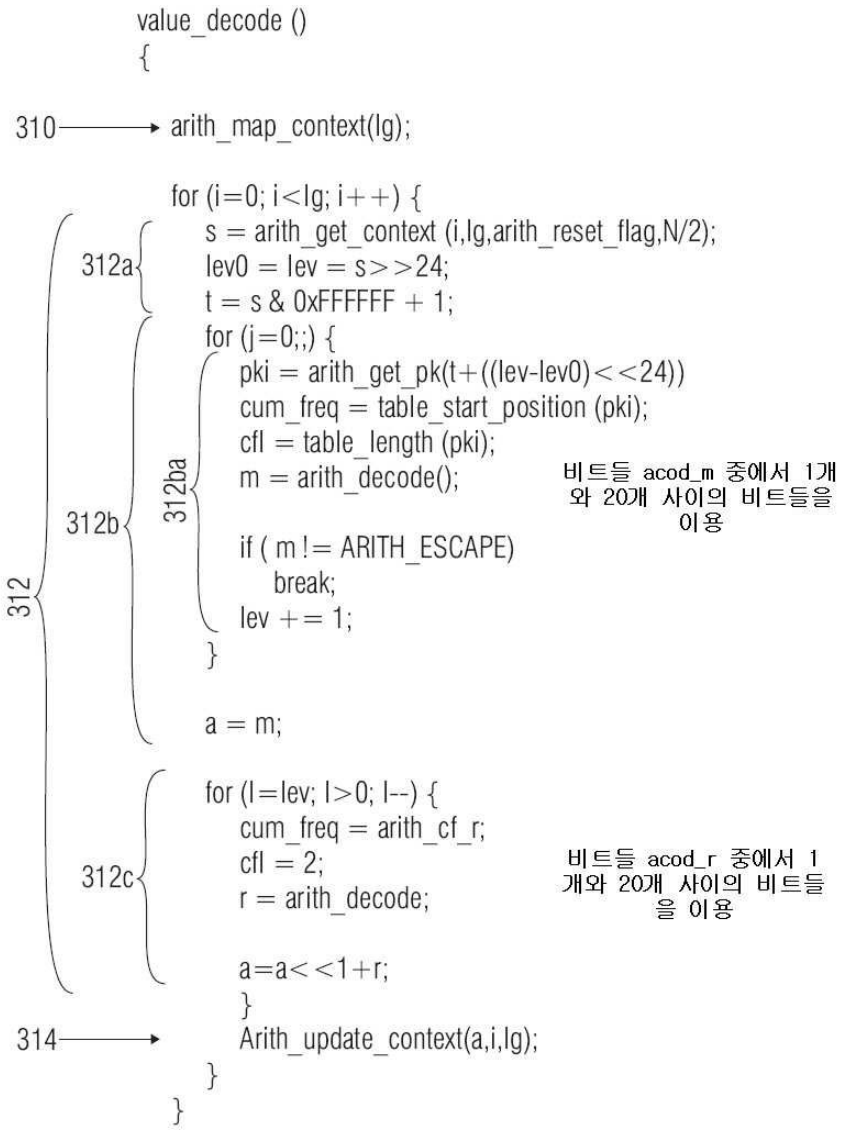
도면2a


230
산술 디코더
스펙트럼 값들의 튜플의 최상위 비트플레인의 디코딩된 값들
284
222
인코딩된 주파수 영역 오디오 표현
예컨대, 산술적으로 코딩된 스펙트럼 데이터 acod_m
최상위 비트플레인 결정기
286
하위 비트플레인들 정보의 갯수
택일적 사항: 하위 비트플레인 결정기
210
비트스트림
(인코딩된 오디오 정보)
택일적 사항: 비트스트림 페이로드 디포맷터
acod_r (스펙트럼 값들의 산술적으로 인코딩된 표현)
288
290
스펙트럼 값들의 튜플의 하나 이상의 하위 비트플레인들의 디코딩된 값들
297
누적 도수 테이블
누적 도수 테이블 선택기
296
220
298
상태 인덱스
택일적 사항: 상태 표현
상태 추적기
224
299
오디오 디코더

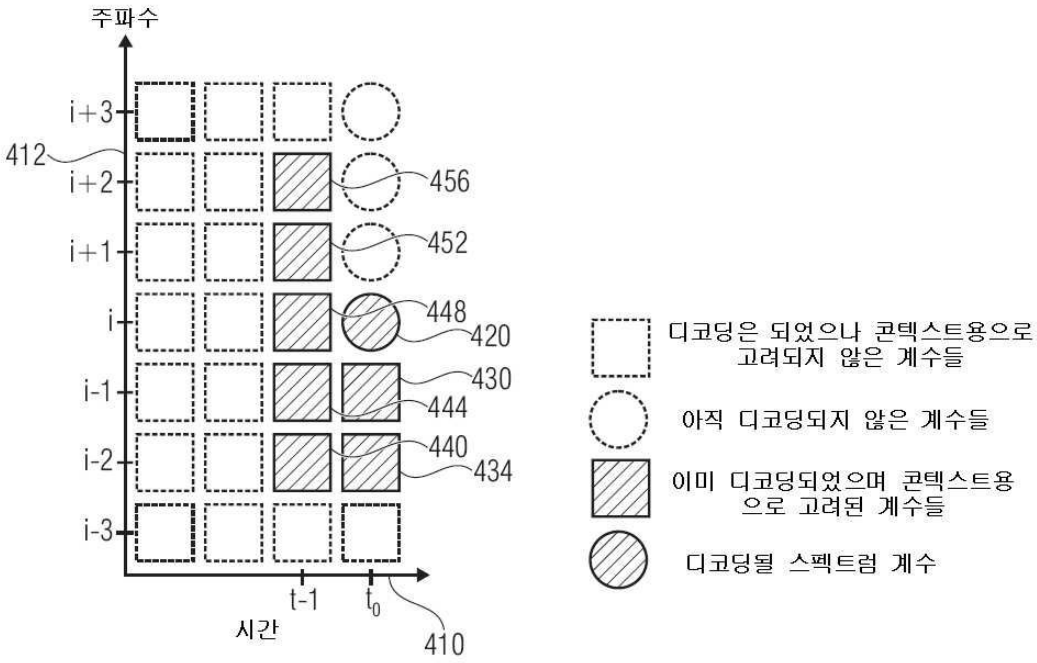
도면2b


200
택일적 사항: 비트플레인 결합기
역 양자화되고 리스케일링된 주파수 영역 오디오 표현
232
252
260
262
270
택일적 사항: 비트플레인 결합기
택일적 사항: 역 양자화된 리스케일러
택일적 사항: 스펙트럼 사전처리기
주파수 영역-시간 영역 신호 변환기 (신호 컨버터)
택일적 사항: 시간 영역 후처리
292
240
242
250
디코딩된 주파수 영역 오디오 표현
예를 들어, 역 변형 이산 코사인 변환 및 윈도우잉
인코딩된 오디오 정보의 시간 영역 표현
디코딩된 오디오 정보
212
오디오 디코더

도면3


value_decode ()
{
310
arith_map_context(lg);
for (i=0; i<lg; i++) {
312
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();
비트들 acod_m 중에서 1개와 20개 사이의 비트들을 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;
비트들 acod_r 중에서 1개와 20개 사이의 비트들을 이용
a=a<<1+r;
}
314
Arith_update_context(a,i,lg);
}
}

도면4

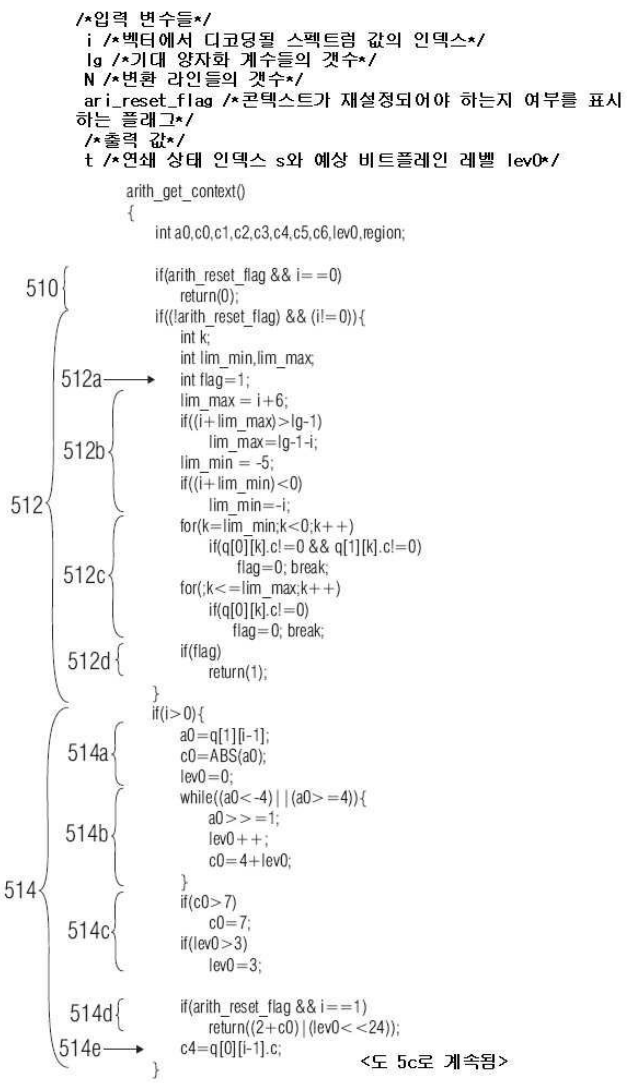


상태 계산을 위한 콘텍스트

도면5a

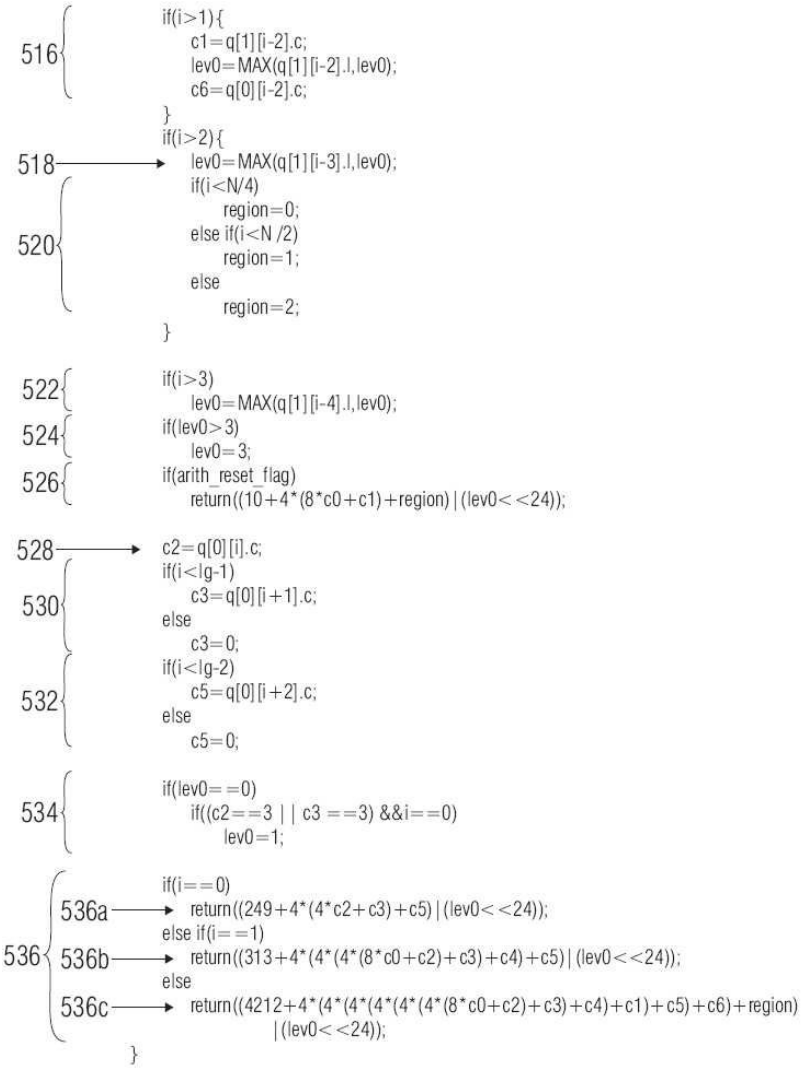
```
/*입력 변수들*/
lg /*프레임에서 디코딩될 스펙트럼 계수들의 갯수*/
previous_lg /*이전 프레임의 스펙트럼 라인들의 이전 갯수*/
arith_map_context()
{
    v=w=0

    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

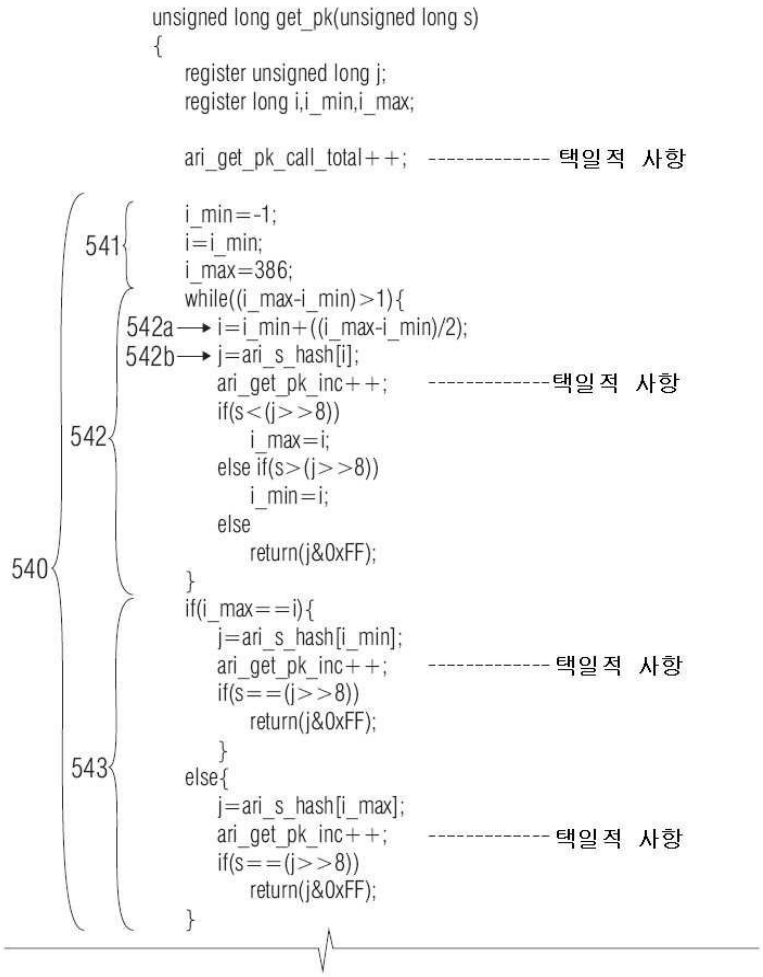

/*입력 변수들*/
i /*벡터에서 디코딩될 스펙트럼 값의 인덱스*/
lg /*기대 양자화 계수들의 갯수*/
N /*변환 라인들의 갯수*/
ari_reset_flag /*콘텍스트가 재설정되어야 하는지 여부를 표시하는 플래그*/
/*출력 값*/
t /*연쇄 상태 인덱스 s와 예상 비트플레인 레벨 lev0*/
arith_get_context()
{
int a0,c0,c1,c2,c3,c4,c5,c6,lev0,region;
510
if(arith_reset_flag && i==0)
return(0);
if((!arith_reset_flag) && (i!=0)){
int k;
int lim_min,lim_max;
512a
int flag=1;
lim_max = i+6;
512b
if((i+lim_max)>lg-1)
lim_max=lg-1-i;
lim_min = -5;
if((i+lim_min)<0)
512
lim_min=-i;
for(k=lim_min;k<0;k++)
512c
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;
512d
if(flag)
return(1);
}
if(i>0){
514a
a0=q[1][i-1];
c0=ABS(a0);
lev0=0;
514b
while((a0<-4)||(a0>=4)){
a0>>=1;
lev0++;
c0=4+lev0;
514
}
514c
if(c0>7)
c0=7;
if(lev0>3)
lev0=3;
514d
if(arith_reset_flag && i==1)
return((2+c0)|(lev0<<24));
514e
c4=q[0][i-1].c;
}
<도 5c로 계속됨>

도면5c

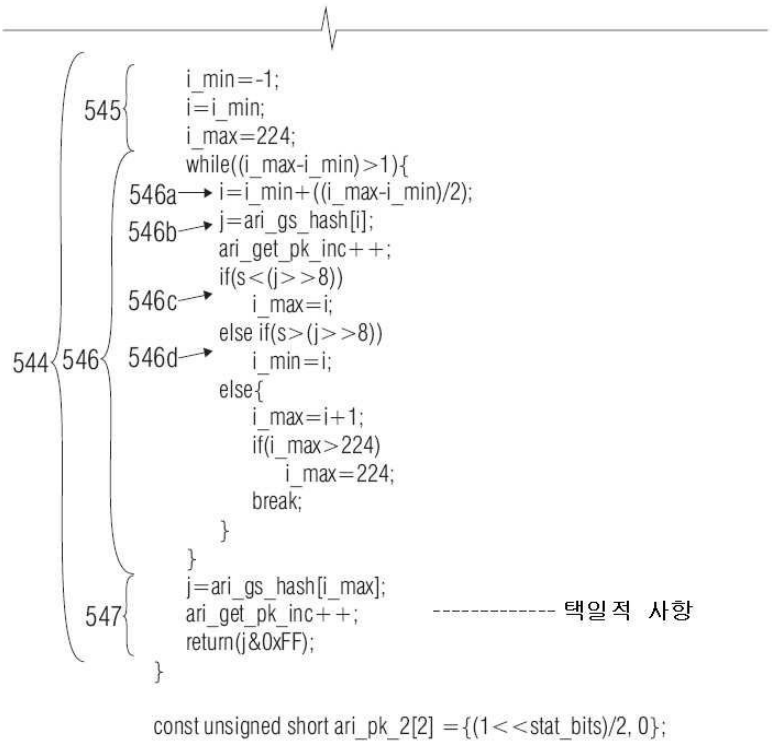
<도 5b로부터 계속됨>



도면5da


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

도면5db



도면5e

```
/*입력 변수들*/
s /*콘텍스트의 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;
}
```

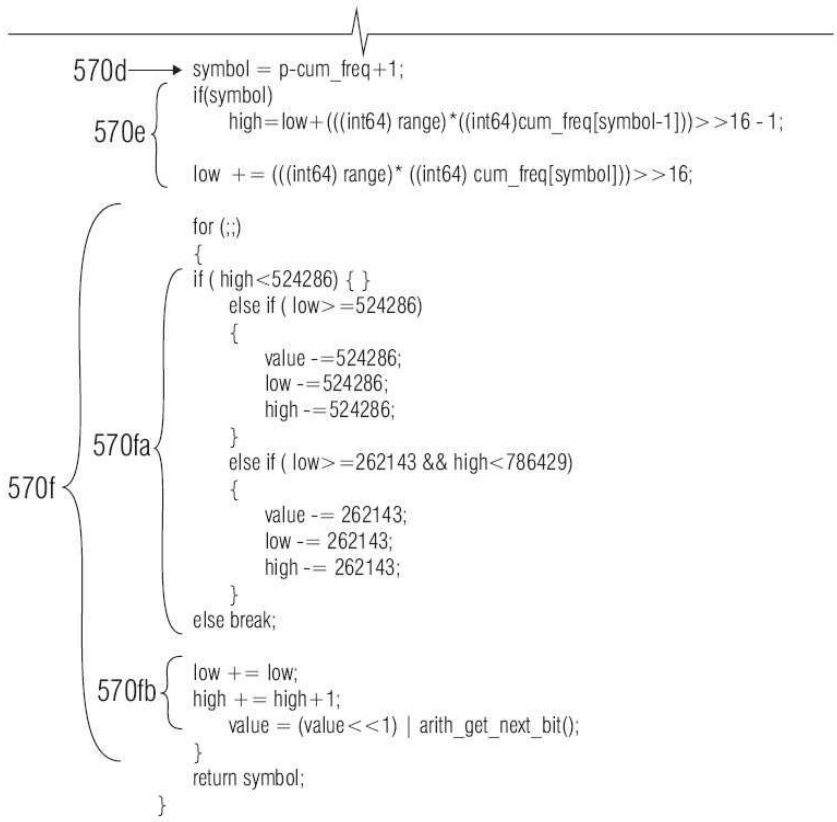
도면5f

```
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);
}
```

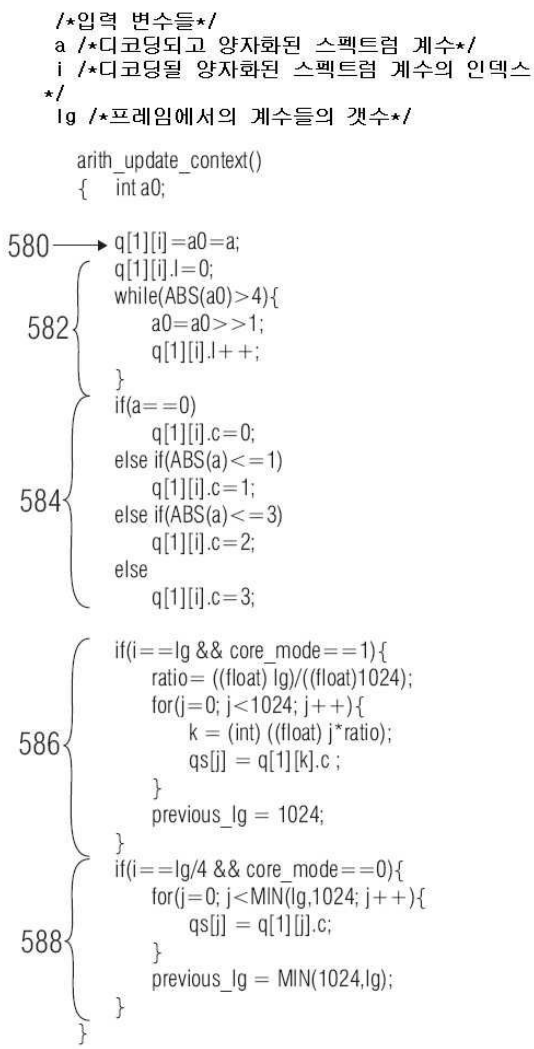
도면5ga

```
/*헬퍼 함수들*/
/*시퀀스의 첫번째 심볼인 경우 TRUE를 반환시키고, 그렇
지 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++)
        {
570a    value = (val<<1) | arith_get_next_bit();
        }
        low=0;
        high=1048575;
    }

570b{ range = high-low+1;
    cum =((((int64) (value-low+1))<<16)-((int64) 1))/((int64) range);
    p = cum_freq-1;

570c{ do
    {
        q=p+(cfl>>1);
        if ( *q > cum ) {p=q; cfl++; }
        cfl>>=1;
    }
        while ( cfl>1 );
```

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;
582
while(ABS(a0)>4){
a0=a0>>1;
q[1][i].l++;
}
584
if(a==0)
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;
586
if(i==lg && core_mode==1){
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;
}
588
if(i==lg/4 && core_mode==0){
for(j=0; j<MIN(lg,1024; j++){
qs[j] = q[1][j].c;
}
previous_lg = MIN(1024,lg);
}
}

도면5i

a	디코딩될 양자화된 계수
m	디코딩될 양자화된 스펙트럼 계수의 최상위 2비트 와이즈 플레인
r	디코딩될 양자화된 스펙트럼 계수의 최상위 2 비트 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
if (core_mode == 1) {
lpd_channel_stream();
}
else {
fd_channel_stream();
}
}
	1	uimsbf

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	VlclbfVlclbf
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

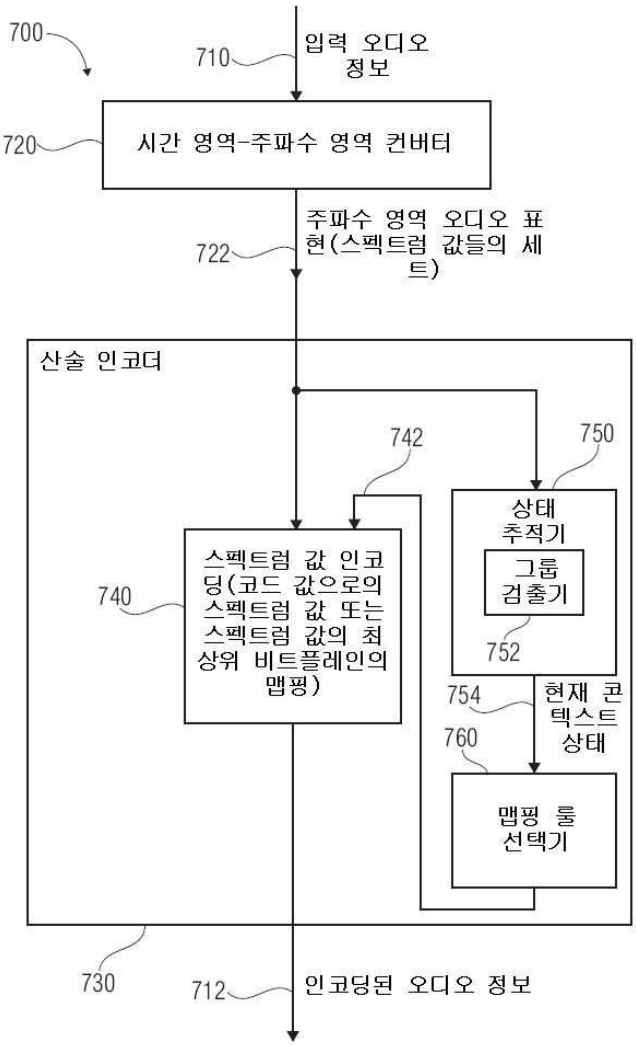
정의들

arith_data()	스펙트럼 무잡음 코더 데이터를 디코딩하기 위한 데이터 엘리먼트
arith_reset_flag	스펙트럼 무잡음 콘텍스트가 재설정되어야 하는 지를 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

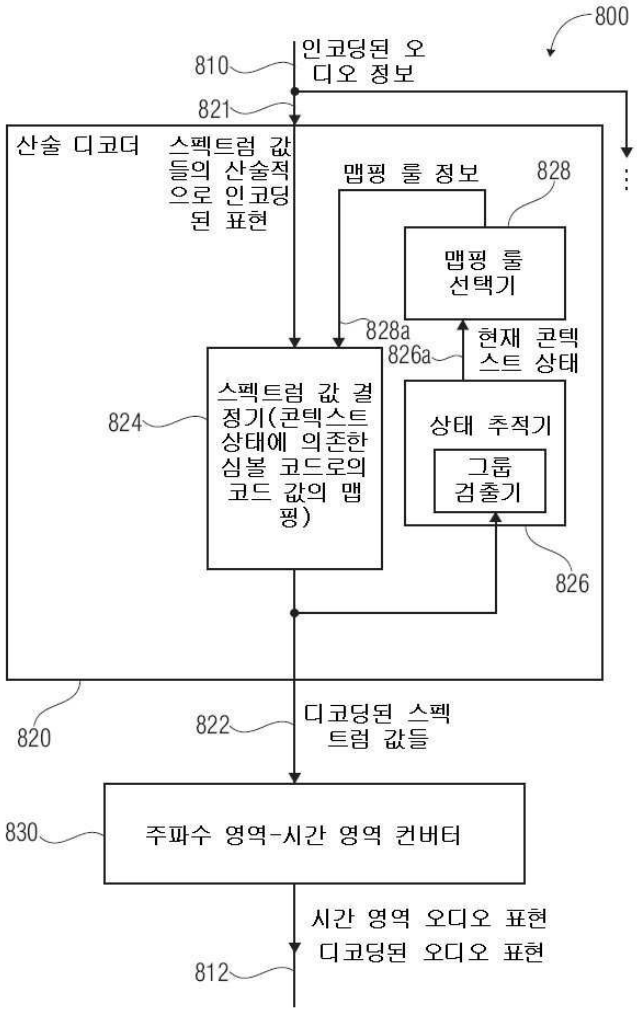
도움 구성요소들

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
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

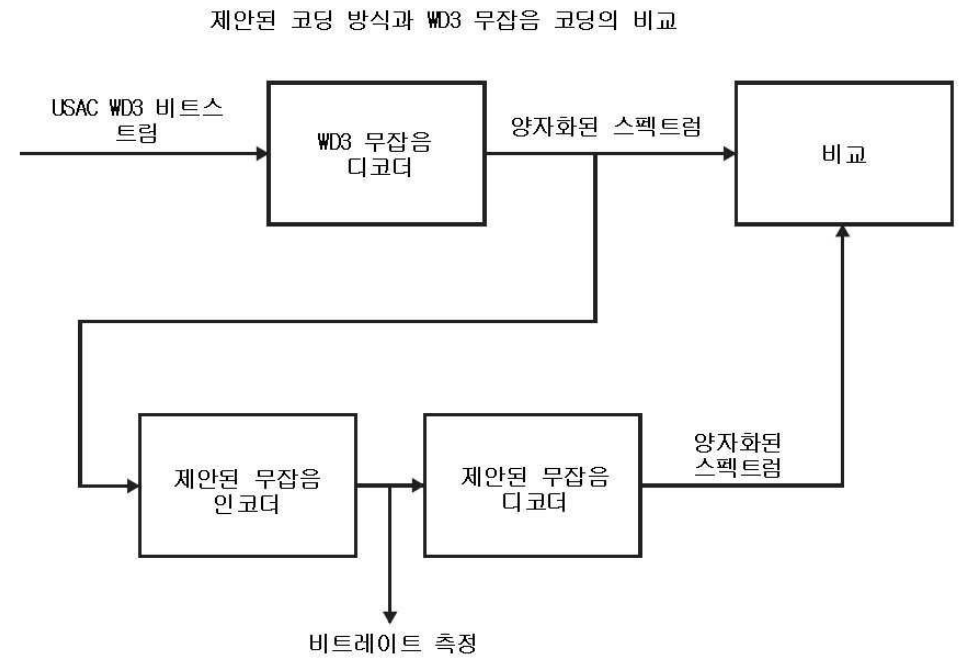
도면7


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

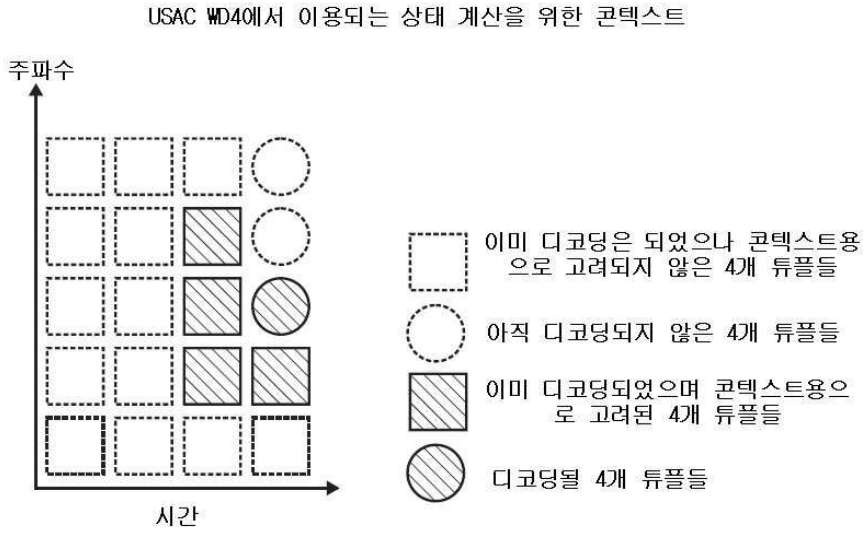
도면8


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

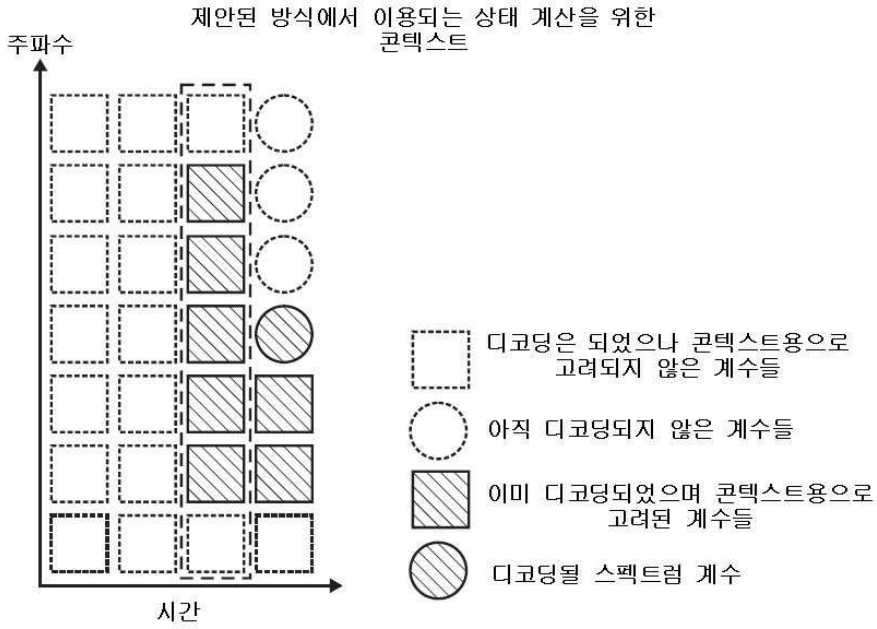
도면9


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

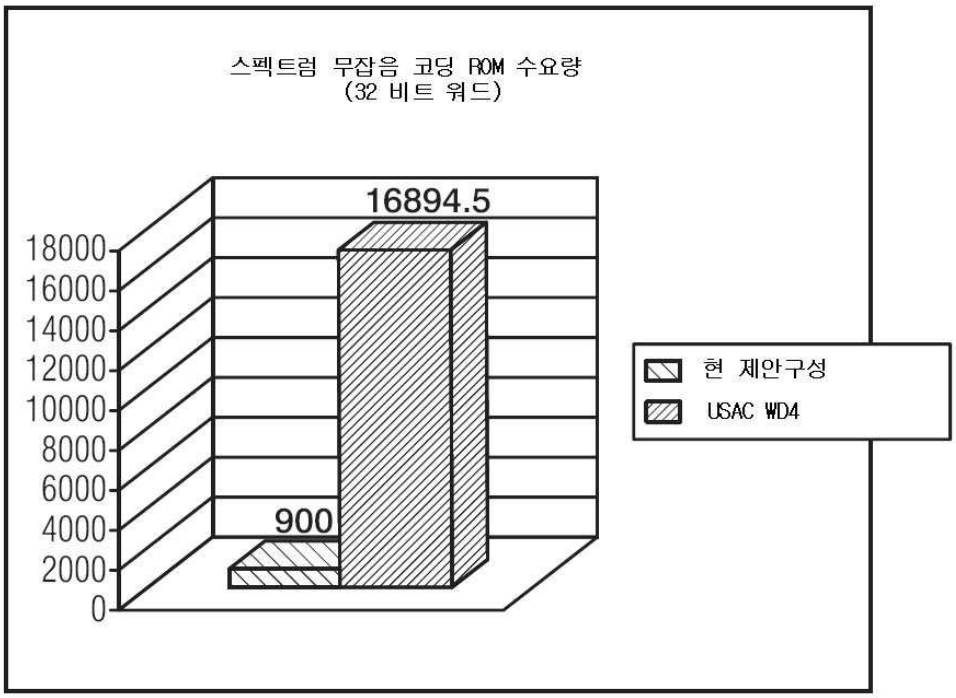
도면11b

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

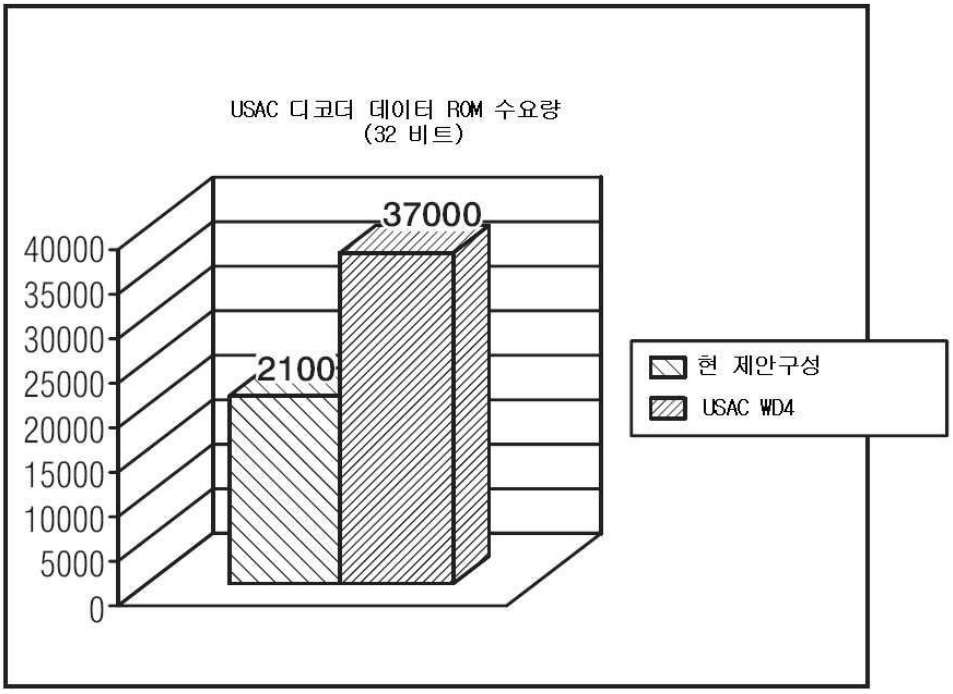
도면12a

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,0x0015B619,0x0015B714,0x00207510,0x00207612,0x00207713,0x0020791D,
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},
{65490,65339,64638,58133, 7463,  973,  270,  125,    0},
{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}
};
```

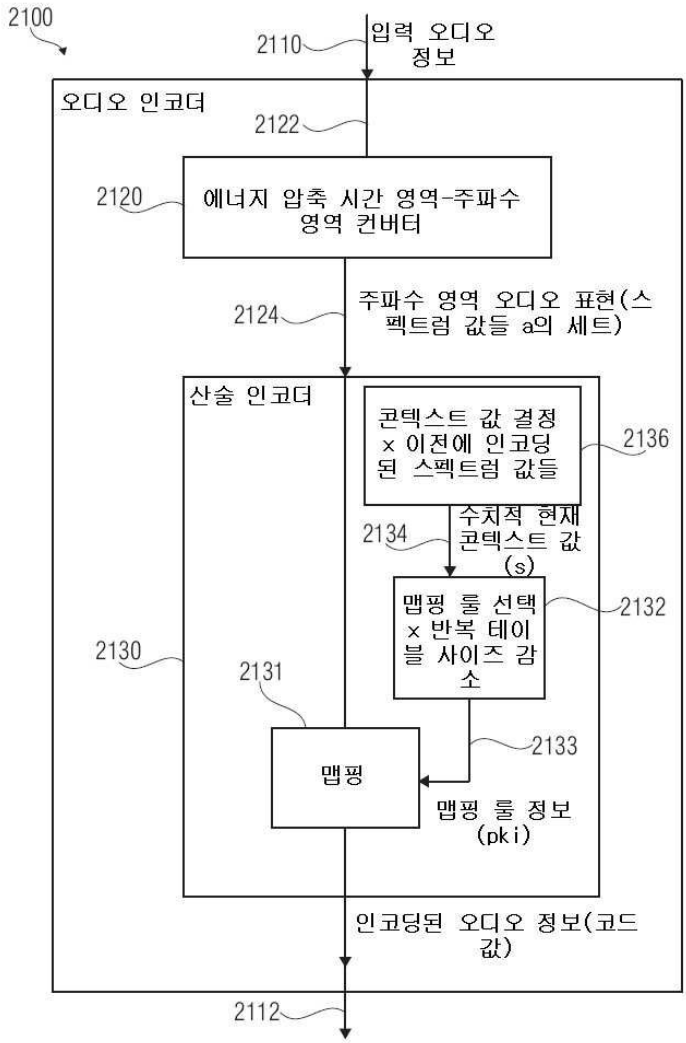
도면20a

```
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,
```

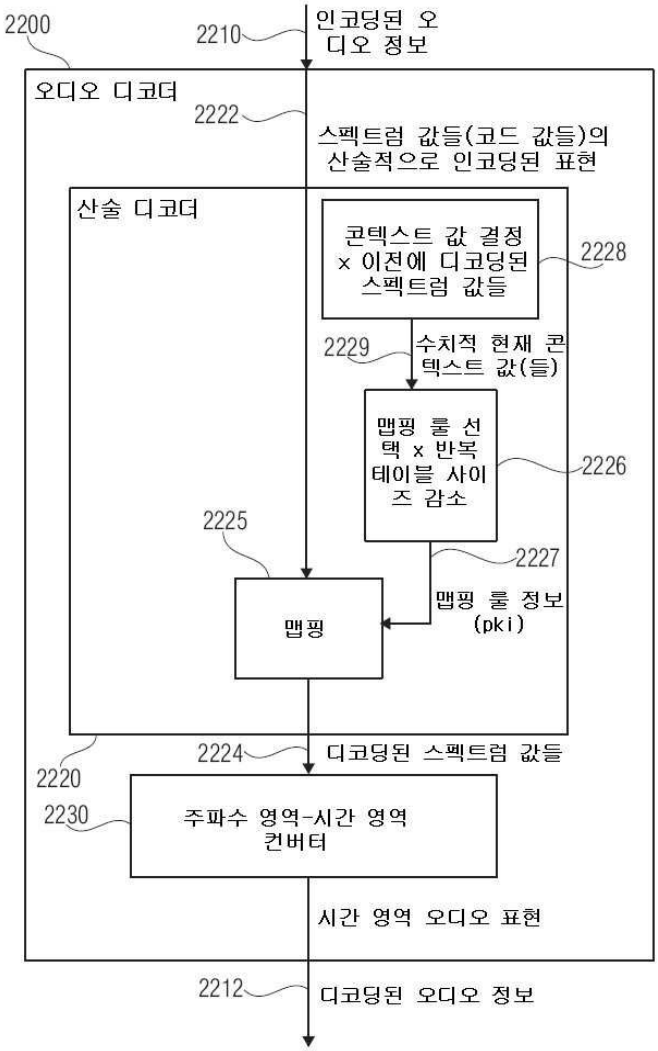
도면20b

```
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
};
```

도면21


2100
2110
입력 오디오 정보
오디오 인코더
2122
2120
에너지 압축 시간 영역-주파수 영역 컨버터
2124
주파수 영역 오디오 표현(스펙트럼 값들 a의 세트)
산술 인코더
콘텍스트 값 결정 × 이전에 인코딩 된 스펙트럼 값들
2136
2134
수치적 현재 콘텍스트 값 (s)
맵핑 룰 선택 × 반복 테이블 사이즈 감소
2132
2130
2131
2133
맵핑
맵핑 룰 정보 (pki)
인코딩된 오디오 정보(코드 값)
2112

도면22


2200
2210
인코딩된 오디오 정보
오디오 디코더
2222
스펙트럼 값들(코드 값들)의 산술적으로 인코딩된 표현
산술 디코더
콘텍스트 값 결정 x 이전에 디코딩된 스펙트럼 값들
2228
2229
수치적 현재 콘텍스트 값(들)
맵핑 룰 선택 x 반복 테이블 사이즈 감소
2226
2225
2227
맵핑
맵핑 룰 정보 (pki)
2224
디코딩된 스펙트럼 값들
2220
2230
주파수 영역-시간 영역 컨버터
시간 영역 오디오 표현
2212
디코딩된 오디오 정보