(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2013년12월04일
(11) 등록번호 10-1336051
(24) 등록일자 2013년11월27일

(51) 국제특허분류(Int. Cl.)
G10L 19/00 (2006.01)
(21) 출원번호 10-2012-7021154
(22) 출원일자(국제) 2011년01월11일
심사청구일자 2012년08월10일
(85) 번역문제출일자 2012년08월10일
(65) 공개번호 10-2012-0109621
(43) 공개일자 2012년10월08일
(86) 국제출원번호 PCT/EP2011/050273
(87) 국제공개번호 WO 2011/086066
국제공개일자 2011년07월21일
(30) 우선권주장
61/294,357 2010년01월12일 미국(US)
(56) 선행기술조사문헌
OLIVER WUEBBOLT: "Spectral Noi sel ess 1 , 16-19 Coding CE: Thomson proposal " , 90. MPEG MEETING; 26-10-2009 - 30-10-2009 ; XIAN ; (MOTION PICTURE EXPERT GROUP OR ISO/I EC JTC1/SC29/WG11) ,23 Oct.
전체 청구항 수 : 총 19 항

(73) 특허권자
프라운호퍼 게젤샤프트 쭈르 푀르데룽 데어 안겐반텐 포르슝 에. 베.
독일 80686 뮌헨 한자슈트라쎄 27 체
(72) 발명자
수바라만, 비네쉬
독일 91054 에를랑겐 바이로이트슈트라쎄 35에이
푸호스, 구일라우메
독일 91058 에를랑겐 푸르트 슈트라쎄 17
(뒷면에 계속)
(74) 대리인
김수진, 윤의섭

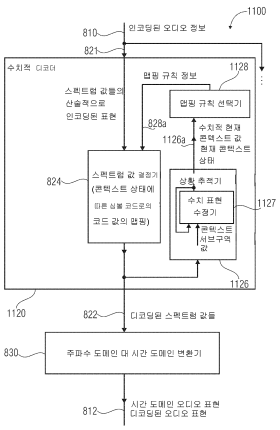
심사관 : 이수철

(54) 발명의 명칭 **오디오 인코더, 오디오 디코더, 오디오 정보 인코딩 방법, 오디오 정보 디코딩 방법, 및 수치적 이전 콘텍스트 값의 수치 표현의 수정을 이용하는 컴퓨터 프로그램**

(57) 요 약

인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더는 스펙트럼 값들의 산술적으로 인코딩된 표현에 기초하여 복수의 디코딩된 스펙트럼 값들을 제공하기 위한 산술 디코더, 및 디코딩된 오디오 정보를 획득하기 위해 디코딩된 스펙트럼 값들을 이용하여 시간 도메인 오디오 표현을 제공하기 위한 주파수 도메인 대 시간 도메인 변환기를 포함한다. 산술 디코더는 수치적 현재 콘텍스트 값에 의해 기술된 콘텍스트 상태에 따라 심볼 코드로의 코드 값의 맵핑을 기술하는 맵핑 규칙을 선택하도록 구성된다. 산술 디코더는 복수의 이전에 디코딩된 스펙트럼 값들에 따라 수치적 현재 콘텍스트 값을 결정하도록 구성된다. 산술 디코더는, 디코딩되는 하나 이상의 스펙트럼 값들과 연관된 콘텍스트 값을 기술하는 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 콘텍스트 서브구역 값에 따라, 하나 이상의 이전에 디코딩된 스펙트럼 값들과 연관된 콘텍스트 상태를 기술하는, 수치적 이전 콘텍스트 값의 수치 표현을 수정하도록 또한 구성된다. 오디오 인코더는 유사한 구상을 이용한다.

대 표 도 - 도11



(72) 발명자

물트루스, 마르쿠스

독일 90469 뉘른베르크 에츠라우프벡그 7

레텔바흐, 리콜라우스

독일 90427 뉘른베르크 스페싸트슈트라쎄 38

가이어, 마크

독일 91058 에를랑겐 폴켄나우어슈트라쎄 3

바이스, 올리버

독일 90459 뉘른베르크 페터-헨라인-슈트라쎄 45

그리벨, 크리스티안

독일 90402 뉘른베르크 오스텐트슈트라쎄 44

밤볼트, 패트릭

독일 91448 엠스키르크헨 마우스도르프 50

특허청구의 범위

청구항 1

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

디코딩된 오디오 정보(212;812)를 획득하기 위해, 상기 디코딩된 스펙트럼 값들(232;822)을 이용하여 시간 도메인 오디오 표현(262;812)을 제공하기 위한 주파수 도메인 대 시간 도메인 변환기(260;830);

를 포함하되,

상기 산술 디코더(230;820)는 수치적 현재 콘텍스트(context) 값(c)에 의해 기술된 콘텍스트 상태에 따라 상기 디코딩된 스펙트럼 값들 중에서 하나 이상 또는 상기 디코딩된 스펙트럼 값들 중에서 하나 이상의 적어도 일부를 표현하는 심볼 코드(symbol)로의, 스펙트럼 값들의 상기 산술적으로 인코딩된 표현(222;821)의 코드 값(value)의 맵핑을 기술하는 맵핑 규칙(297;cum_freq[])을 선택하도록 구성되고;

상기 산술 디코더(230;820)는 수치적 이전 콘텍스트 값 및 복수의 이전에 디코딩된 스펙트럼 값들에 따라 상기 수치적 현재 콘텍스트 값(c)을 결정하도록 구성되며,

상기 산술 디코더는, 디코딩되는 하나 이상의 스펙트럼 값들의 디코딩을 위한 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 콘텍스트의 서브구역을 기술하는 콘텍스트 서브구역 값(q[][])에 따라, 하나 이상의 이전에 디코딩된 스펙트럼 값들의 디코딩을 위한 콘텍스트 상태를 기술하는, 상기 수치적 이전의 콘텍스트 값의 수치 표현을 수정하도록 구성되는 것을 특징으로 하는 인코딩된 오디오 정보(210;810)에 기초하여 디코딩된 오디오 정보(212;812)를 제공하기 위한 오디오 디코더(200;800).

청구항 2

청구항 1에 있어서,

상기 산술 디코더는 각각 다른 수치적 가중치들을 갖는 상기 수치 표현의 부분들이 각각 다른 콘텍스트 서브구역 값들(q[][])에 의해 결정되도록 상기 수치적 현재 콘텍스트 값의 수치 표현을 제공하기 위해 구성되는 것을 특징으로 하는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더.

청구항 3

청구항 1에 있어서,

상기 수치 표현은 단일 수치적 현재 콘텍스트 값(c)의 이진 수치 표현이고;

상기 이진 수치 표현의 비트들의 제1 서브셋(subset)은 하나 이상의 이전에 디코딩된 스펙트럼 값들과 연관된 제1 콘텍스트 서브구역 값에 의해 결정되고;

상기 이진 수치 표현의 비트들의 제2 서브셋은 하나 이상의 이전에 디코딩된 스펙트럼 값들과 연관된 제2 콘텍스트 서브구역 값에 의해 결정되며, 비트들의 상기 제1 서브셋의 비트들은 비트들의 상기 제2 서브셋의 비트들과는 다른 수치적 가중치를 포함하는 것을 특징으로 하는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더.

청구항 4

청구항 1에 있어서,

상기 산술 디코더는, 상기 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 상기 수치적 이전 콘텍스트 값의 도출을 위해 고려되지 않은 콘텍스트 서브구역 값에 따라, 상기 수치적 이전 콘텍스트 값들의 수치 표현의

정보 비트들, 또는 상기 수치적 이전 콘텍스트 값의 수치 표현의 비트 이동된 버전의 비트 방식 마스킹된(bit-wise masked) 서브셋을 수정하도록 구성되는 것을 특징으로 하는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더.

청구항 5

청구항 1에 있어서,

상기 산술 디코더는, 상기 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 각각 다른 콘텍스트 서브구역 값들과 연관된 비트들의 서브셋들의 수치적 가중치들이 수정되도록, 상기 수치적 이전 콘텍스트 값의 수치 표현을 비트 이동하기 위해 구성되는 것을 특징으로 하는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더.

청구항 6

청구항 5에 있어서,

상기 산술 디코더는, 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 콘텍스트 서브구역 값과 연관되는, 비트들의 서브셋이 상기 수치 표현으로부터 삭제되도록, 상기 수치적 이전 콘텍스트 값의 수치 표현을 비트 이동하기 위해 구성되는 것을 특징으로 하는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더.

청구항 7

청구항 1에 있어서,

상기 산술 디코더는, 상기 이전에 디코딩된 스펙트럼 값들의 디코딩을 위해 고려되고 상기 수치적 현재 콘텍스트 값을 이용하여 디코딩되는 스펙트럼 값들의 디코딩을 위해서는 고려되지 않는 콘텍스트 서브구역들과 연관된 비트들의 하나 이상의 서브셋들을 선택적으로 수정함으로써 상기 수치적 이전 콘텍스트 값의 이진 수치 표현으로부터 상기 수치적 현재 콘텍스트 값의 이진 수치 표현을 도출하기 위해, 콘텍스트 서브구역 값에 따라, 수치적 이전 콘텍스트 값의 이진 수치 표현의 비트들, 또는 수치적 이전 콘텍스트 값의 이진 수치 표현의 비트 이동된 버전의 제1 서브셋을 수정하고, 상기 수치적 이전 콘텍스트 값의 이진 수치 표현의 비트들, 또는 상기 수치적 현재 콘텍스트 값의 이진 수치 표현의 비트 이동된 버전의 제2 서브셋들을 변경되지 않은 채로 두도록 구성되는 것을 특징으로 하는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더.

청구항 8

청구항 1에 있어서,

상기 산술 디코더는, 상기 수치적 현재 콘텍스트 값의 수치 표현의 최하위 비트들의 서브셋이, 콘텍스트 상태가 상기 수치적 현재 콘텍스트 값에 의해 정의되는 스펙트럼 값들의 디코딩을 위해 이용되고, 콘텍스트 상태가 수치적 다음(subsequent) 콘텍스트 값에 의해 정의되는 스펙트럼 값들의 디코딩을 위해서는 이용되지 않는, 콘텍스트 서브구역 값을 기술하도록 상기 수치적 현재 콘텍스트 값의 수치 표현을 제공하기 위해 구성되는 것을 특징으로 하는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더.

청구항 9

청구항 1에 있어서,

상기 산술 디코더는, 상기 수치적 현재 콘텍스트 값이 테이블의 엔트리에 의해 기술된 테이블 콘텍스트 값과 동

일한지 또는 상기 테이블의 엔트리들에 의해 기술된 구간 내에 있는지 여부를 결정하기 위해, 적어도 하나의 테이블을 평가하고, 상기 적어도 하나의 테이블에 대한 평가 결과에 따라 선택된 맵핑 규칙을 기술하는 맵핑 규칙 인덱스 값을 도출하도록 구성되는 것을 특징으로 하는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더.

청구항 10

청구항 1에 있어서,

상기 산술 디코더는 복수의 콘텍스트 서브구역 값들의 합계가 미리 결정된 합계 임계 값보다 더 작은지 또는 미리 결정된 합계 임계 값과 같은지 여부를 검사하고, 검사 결과에 따라 상기 수치적 현재 콘텍스트 값을 선택적으로 수정하도록 구성되는 것을 특징으로 하는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더.

청구항 11

청구항 10에 있어서,

상기 산술 디코더는, 콘텍스트 상태를 이용하여 디코딩되는 상기 하나 이상의 스펙트럼 값들이 상기 수치적 현재 콘텍스트 값에 정의됨으로써 상기 오디오 콘텐츠의 동일한 시간 부분과 연관되고, 상기 수치적 현재 콘텍스트 값에 의해 정의된 콘텍스트 상태를 이용하여 디코딩되는 상기 하나 이상의 스펙트럼 값들보다 더 낮은 주파수들과 연관되는, 복수의 콘텍스트 서브구역 값들의 합계가 미리 결정된 합계 임계 값보다 더 작은지 또는 미리 결정된 합계 임계 값과 같은지 여부를 검사하고, 상기 검사 결과에 따라 상기 수치적 현재 콘텍스트 값을 선택적으로 수정하도록 구성되는 것을 특징으로 하는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더.

청구항 12

청구항 1에 있어서,

상기 산술 디코더는, 제1 복수의 이전에 디코딩된 스펙트럼 값들과 연관된 제1 콘텍스트 서브구역 값을 획득하기 위해 제1 복수의 이전에 디코딩된 스펙트럼 값들의 절대 값들을 합계하고, 제2 복수의 이전에 디코딩된 스펙트럼 값들과 연관된 제2 콘텍스트 서브구역 값을 획득하기 위해 제2 복수의 이전에 디코딩된 스펙트럼 값들의 절대 값들 합계하도록 구성되는 것을 특징으로 하는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더.

청구항 13

청구항 1에 있어서,

상기 산술 디코더는, 상기 수치적 이전 콘텍스트 값의 수치 표현의 정보 비트들의 트루(true) 서브셋을 이용하여 상기 콘텍스트 서브구역 값들이 표현될 수 있도록, 상기 콘텍스트 서브구역 값들을 제한하기 위해 구성되는 것을 특징으로 하는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더.

청구항 14

청구항 1에 있어서,

상기 산술 디코더는,

다음의 알고리즘:

```
c = c>>4;
        if(i<i_max-1)
                c = c + (q[0][i+1]<<12);
        c = (c&0xFFF0);
        if(i>0)
c = c + (q[1][i-1]);
```

을 이용하여, 상기 수치적 이전 콘텍스트 값으로부터 상기 수치적 현재 콘텍스트 값 c를 도출하기 위해, 상기 수치적 이전 콘텍스트 값의 이진 수치 표현 c를 업데이트하도록 구성되되,

c는 상기 알고리즘의 실행 이전의 상기 수치적 이전 콘텍스트 값을, 이진 표현으로, 표현하고, 상기 알고리즘의 실행 이후의 상기 수치적 현재 콘텍스트 값을, 이진 표현으로, 표현하는 변수이며;

">>4"는 "4 비트 만큼 오른쪽으로 이동" 연산을 지칭하며;

i는 상기 수치적 현재 콘텍스트 값을 이용하여 디코딩되는 상기 하나 이상의 스펙트럼 값들의 주파수 인덱스이며;

i_max는 주파수 인덱스들의 전체 숫자를 지칭하며;

q[0][i+1]는 상기 수치적 현재 콘텍스트 값을 이용하여 디코딩되는 하나 이상의 스펙트럼 값들의 주파수들보다 더 높은 주파수들 및 상기 오디오 콘텐츠의 이전 시간 부분에 대한 하나 이상의 이전에 디코딩된 스펙트럼 값들과 연관된 콘텍스트 서브구역 값을 지칭하며;

"<<12"는 "12 비트 만큼 왼쪽으로 이동" 연산을 지칭하며;

"&0xFFF0"은 16진 값 "0xFFF0"을 갖는 부울 AND(Boolean-AND) 연산을 지칭하고;

q[1][i-1]은 상기 수치적 현재 콘텍스트 값을 이용하여 디코딩되는 하나 이상의 스펙트럼 값들의 주파수들보다 더 낮은 주파수 들 및 상기 오디오 콘텐츠의 현재 시간 부분에 대한 하나 이상의 이전에 디코딩된 스펙트럼 값들과 연관된 콘텍스트 서브구역 값을 지칭하는 것을 특징으로 하는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더.

청구항 15

청구항 14에 있어서,

상기 산술 디코더는, 만약

(q[1][i-3]+q[1][i-2]+q[1][i-1]) <5;

이면, 16진 값 0x10000 만큼 c를 증가시켜 상기 수치적 현재 콘텍스트 값의 상기 이진 수치 표현 c를 선택적으로 수정하도록 구성되며,

q[1][i-3], q[1][i-2], 및 q[1][i-1]은, 상기 수치적 현재 콘텍스트 값을 이용하여 디코딩되는 하나 이상의 스펙트럼 값들의 주파수들보다 더 낮은 주파수 들 및 상기 오디오 콘텐츠의 현재 시간 부분에 대한 하나 이상의 이전에 디코딩된 스펙트럼 값들과 각각 연관된, 콘텍스트 서브구역 값들인 것을 특징으로 하는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더.

청구항 16

입력 오디오 정보의 시간 도메인 표현(110;710)에 기초해 주파수 도메인 오디오 표현(132;722)을 제공하여, 상기 주파수 도메인 오디오 표현(132;722)이 스펙트럼 값들의 셋트를 포함하는, 에너지 압축(compacting) 시간 도메인 대 주파수 도메인 변환기(130;720); 및

가변 길이 코드워드(variable length codeword, acod_m, acod_r)를 이용하여 스펙트럼 값(a) 또는 상기 스펙트

럼 값(a)의 전처리된 버전을 인코딩하도록 구성되며, 산술 인코더(170)는, 코드 값(acod_m)에, 하나 이상의 스펙트럼 값들(a, b) 또는 하나 이상의 스펙트럼 값(a, b)의 최상위 비트평면의 값(m)을 맵핑하도록 구성되는, 산술 인코더(170;730);

를 포함하되,

인코딩된 오디오 정보는 복수의 가변 길이 코드워드들을 포함하며,

상기 산술 인코더는, 수치적 현재 콘텍스트 값(c)에 의해 기술된 콘텍스트 상태(s)에 따라, 코드 값으로의, 하나 이상의 스펙트럼 값들 또는 하나 이상의 스펙트럼 값들의 최상위 비트평면 값의 맵핑을 기술하는 맵핑 규칙을 선택하도록 구성되고;

상기 산술 인코더는 수치적 이전 콘텍스트 값 및 복수의 이전에 인코딩된 스펙트럼 값들에 따라 상기 수치적 현재 콘텍스트 값(c)을 결정하도록 구성되며,

상기 산술 인코더는, 인코딩되는 하나 이상의 스펙트럼 값들의 인코딩을 위한 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 콘텍스트의 서브구역을 기술하는 콘텍스트 서브구역 값에 따라, 하나 이상의 이전에 인코딩된 스펙트럼 값들의 인코딩을 위한 콘텍스트 상태를 기술하는, 상기 수치적 이전 콘텍스트 값의 수치 표현(c)을 수정하도록 구성되는 것을 특징으로 하는 입력된 오디오 정보(110;710)에 기초하여 인코딩된 오디오 정보를 제공하기 위한 오디오 인코더(100;700).

청구항 17

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

디코딩된 오디오 정보를 획득하기 위하여, 상기 디코딩된 스펙트럼 값들을 이용하여 시간 도메인 오디오 표현을 제공하는 단계;

를 포함하되,

상기 복수의 디코딩된 스펙트럼 값들을 제공하는 단계는, 수치적 현재 콘텍스트 값(c)에 의해 기술된 콘텍스트 상태에 따라 상기 디코딩된 스펙트럼 값들 중에서 하나 이상, 또는 상기 디코딩된 스펙트럼 값들 중에서 하나 이상의 적어도 일부를 표현하는 심볼 코드(symbol)로의, 스펙트럼 값들의 상기 산술적으로 인코딩된 표현(222;821)의 코드 값(acod_m; value)의 맵핑을 기술하는 맵핑 규칙을 선택하는 단계를 포함하고;

상기 수치적 현재 콘텍스트 값(c)은 수치적 이전 콘텍스트 값 및 복수의 이전에 디코딩된 스펙트럼 값들에 따라 결정되며,

하나 이상의 이전에 디코딩된 스펙트럼 값들의 디코딩을 위한 콘텍스트 상태를 기술하는, 상기 수치적 이전 콘텍스트 값의 수치 표현은, 디코딩되는 하나 이상의 스펙트럼 값들의 디코딩을 위한 콘텍스트 상태를 기술하는, 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 콘텍스트의 서브구역을 기술하는 콘텍스트 서브구역 값에 따라 수정되는 것을 특징으로 하는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 방법.

청구항 18

주파수 도메인 오디오 표현이 스펙트럼 값들의 셋트를 포함하도록, 에너지 압축 시간 도메인 대 주파수 도메인 변환을 이용하여 입력된 오디오 정보의 시간 도메인 표현에 기초하여 주파수 도메인 오디오 표현을 제공하는 단계; 및

스펙트럼 값 또는 스펙트럼 값의 최상위 비트평면의 값이 코드 값에 맵핑되되, 가변 길이 코드워드를 이용하여, 스펙트럼 값 또는 상기 스펙트럼 값의 전처리된 버전을 산술적으로 인코딩하는 단계;

를 포함하되,

코드 값으로의, 하나 이상의 스펙트럼 값들 또는 하나 이상의 스펙트럼 값들의 최상위 비트평면의 맵핑을 기술

하는 맵핑 규칙은 수치적 현재 콘텍스트 값(c)에 의해 기술된 콘텍스트 상태에 따라 선택되고;

상기 수치적 현재 콘텍스트 값(c)은 수치적 이전 콘텍스트 값 및 복수의 이전에 인코딩된 스펙트럼 값들에 따라 결정되며;

하나 이상의 이전에 디코딩된 스펙트럼 값들의 인코딩을 위한 콘텍스트 상태를 기술하는, 상기 수치적 이전 콘텍스트 값의 수치 표현은, 인코딩되는 하나 이상의 스펙트럼 값들의 인코딩을 위한 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 콘텍스트의 서브구역을 기술하는 콘텍스트 서브구역 값에 따라 수정되며;

인코딩된 오디오 정보는 복수의 가변 길이 코드워드들을 포함하는 것을 특징으로 하는 입력된 오디오 정보에 기초하여 인코딩된 오디오 정보를 제공하기 위한 방법.

청구항 19

컴퓨터 프로그램이 컴퓨터에서 구동할 때, 청구항 17 또는 18에 따른 방법을 수행하기 위한 컴퓨터 프로그램이 저장된 컴퓨터 판독가능한 기록 매체.

명 세 서

기 술 분 야

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

[0002] 본 발명에 따른 실시예들은, 예를 들어, 이른바 통합 음성 오디오 코더(unified-speech-and-audio coder, USAC)와 같은 오디오 인코더 또는 디코더에서 이용될 수 있는 개선된 스펙트럼 무잡음 코딩(spectral noiseless coding)에 관한 것이다.

배 경 기 술

[0003] 다음에서, 본 발명과 그 장점들에 대한 이해를 용이하게 하기 위해 본 발명의 배경기술이 간단히 설명될 것이다. 지난 10년 동안, 좋은 비트율 효율성으로 오디오 콘텐츠를 디지털로 저장하고 분배하는 가능성을 만드는데 많은 노력이 기울여졌다. 이러한 행보에서 한 가지 중요한 성취는 국제 표준 ISO/IEC 14496-3에서의 정의이다. 이 표준의 제3장은 오디오 콘텐츠 인코딩 및 디코딩에 관한 것이고, 제3장의 제4절은 일반적인 오디오 코딩에 관한 것이다. ISO/IEC 14496 제3장 제4절은 일반적인 오디오 콘텐츠의 인코딩 및 디코딩에 대한 구상을 정의한다. 또한, 품질을 개선시키고/개선시키거나 요구되는 비트율을 감소시키기 위해 추가적인 개선안들이 제안되어 왔다.

[0004] 상기 표준에서 기술된 구상에 따르면, 시간 도메인 오디오 신호가 시간 주파수 표현으로 변환된다. 시간 도메인에서 시간 주파수 도메인으로의 전환은 일반적으로 시간 도메인 샘플들의 "프레임들"이라고도 불리는 전환 블록들을 이용하여 수행된다. 예를 들어, 반 프레임만큼 이동되는(shift) 중첩(overlap) 프레임들을 이용하는 것이 유리하다고 확인됐는데, 중첩이 부작용들(artifacts)을 효과적으로 방지하는(또는 적어도 감소시키는) 것을 가능하게 하기 때문이다. 또한, 일시적으로 제한된 프레임들에 대한 이러한 처리에서 비롯되는 부작용들을 방지하기 위해 윈도윙(windowing)이 수행되어야 하는 것으로 확인됐다.

[0005] 시간 도메인에서 시간 주파수 도메인으로 입력된 오디오 신호의 윈도윙된 부분을 전환함으로써, 많은 경우에 에너지 압축이 얻어져, 몇몇 스펙트럼 값들이 복수의 다른 스펙트럼 값들보다 상당히 더 큰 크기를 포함한다. 이

에 따라, 많은 경우에, 스펙트럼 값들의 평균 크기보다 상당히 위인 크기를 갖는 비교적 소수의 스펙트럼 값들이 있다. 에너지 압축을 야기하는 시간 도메인 대 시간 주파수 도메인 전환의 일반적인 예는 이른바 변형 이산 코사인 변환(modified-discrete-cosine-transform, MDCT)이다.

[0006] 스펙트럼 값들은 심리 음향 모델에 따라 종종 스케일링(scale)되고 양자화되어, 양자화 오류들이 심리 음향적으로 더 중요한 스펙트럼 값들에 대해서는 비교적 더 작고, 심리 음향적으로 덜 중요한 스펙트럼 값들에 대해서는 비교적 더 크다. 스케일링되고 양자화된 스펙트럼 값들은 그 비트율 효율적인 표현을 제공하기 위해 인코딩된다.

[0007] 예를 들어, 양자화된 스펙트럼 계수들에 대한 이른바 허프만 코딩(Huffman coding)의 사용이 국제 표준 ISO/IEC 14496-3:2005(E) 제3장 제4절에 기술된다.

[0008] 그러나, 스펙트럼 값들의 코딩 품질은 요구되는 비트율에 중요한 영향을 갖는 것으로 확인됐다. 또한, 휴대용 소비자 장치로 종종 구현되고, 그래서 저렴하고 저전력을 소비해야 하는 오디오 디코더의 복잡도는 스펙트럼 값들을 인코딩하는데 이용된 코딩에 따르는 것으로 확인됐다.

발명의 내용

해결하려는 과제

[0009] 이러한 형세를 고려하면, 비트율 효율성과 자원 효율성 사이의 개선된 균형(trade-off)을 제공하는 오디오 콘텐츠의 인코딩 및 디코딩을 위한 구상에 대한 필요가 있다.

과제의 해결 수단

[0010] 본 발명에 따른 일 실시예는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 오디오 디코더를 창출한다. 오디오 디코더는 스펙트럼 값들의 산술적으로 인코딩된 표현에 기초하여 복수의 디코딩된 스펙트럼 값들을 제공하기 위한 산술 디코더(arithmetic decoder)를 포함한다. 오디오 디코더는 또한, 디코딩된 오디오 정보를 획득하기 위해, 디코딩된 스펙트럼 값들을 이용하여 시간 도메인 오디오 표현을 제공하기 위한 주파수 도메인 대 시간 도메인 변환기(converter)를 포함한다. 산술 디코더는 수치적 현재 콘텍스트 값에 의해 기술된 콘텍스트 상태(context state)에 따라 (심볼 코드가 일반적으로 스펙트럼 값 또는 복수의 스펙트럼 값들 또는 스펙트럼 값이나 복수의 스펙트럼 값들의 최상위 비트 평면을 기술하는) 심볼 코드로의 심볼 값의 맵핑을 기술하는 맵핑 규칙(mapping rule)을 선택하기 위해 구성된다. 산술 디코더는 복수의 이전에 디코딩된 스펙트럼 값들에 따라 수치적 현재 콘텍스트 값을 결정하기 위해 구성된다. 산술 디코더는, 디코딩되는 하나 이상의 스펙트럼 값들과 연관된 콘텍스트 상태를 기술하는 (또는, 좀더 정확히, 디코딩되는 상기 하나 이상의 스펙트럼 값들의 디코딩을 위한 콘텍스트 상태를 기술하는) 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 콘텍스트 서브구역 값에 따라, 하나 이상의 이전에 디코딩된 스펙트럼 값들과 연관된 콘텍스트 상태를 기술하는 (또는, 좀더 정확히, 상기 하나 이상의 이전에 디코딩된 스펙트럼 값들의 디코딩을 위한 콘텍스트 상태를 기술하는), 수치적 이전 콘텍스트 값의 수치 표현을 수정하도록 구성된다.

[0011] 본 발명에 따른 이 실시예는, 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 콘텍스트 서브구역 값에 따라 수치적 이전 콘텍스트 값의 수치 표현을 수정하는 것이 계산상으로 매우 효율적이라는 결과에 기초하는데, 수치적 현재 콘텍스트 값의 전적인 재계산이 회피될 수 있기 때문이다. 차라리, 계산 노력을 비교적 작게 유지하기 위해 수치적 이전의 콘텍스트 값과 수치적 현재 콘텍스트 값 사이의 연관성이 활용될 수 있다. 수치적 이전 콘텍스트 값의 수치 표현의 재스케일링의 결합, 수치적 이전 콘텍스트 값 또는 수치적 이전 콘텍스트 값의 처리된 수치 표현에 콘텍스트 서브구역 값 또는 (예를 들어, 콘텍스트 서브구역 값의 비트 이동된 버전과 같은)

그것으로부터 도출된 값을 추가, 콘텍스트 서브구역 값에 따라 수치적 이전 콘텍스트 값의 (전체 수치 표현 대신에) 수치 표현의 일부분의 대체, 등을 포함하는, 수치적 이전 콘텍스트 값의 수치 표현의 수정에 대한 많은 다양한 가능성들이 존재하는 것으로 확인됐다. 그러므로, (아마도, 이동된 버전에서) 수치적 이전 콘텍스트 값의 수치 표현의 적어도 일부를 유지하는 것은 수치적 콘텍스트 값의 업데이트를 위한 계산상의 노력을 상당히 감소시키는 것을 가능하게 한다.

[0012] 일 바람직한 실시예에서, 산술 디코더는 각각 다른 수치적 가중치들을 갖는 수치 표현의 부분들이 각각 다른 콘텍스트 서브구역 값들에 의해 결정되도록 수치적 현재 콘텍스트 값의 수치 표현을 제공하기 위해 구성된다. 이에 따라, 수치적 이전 콘텍스트 값으로부터 수치적 현재 콘텍스트 값을 도출하기 위해, 정보 손실을 하지 않으며, 수치적 콘텍스트 값의 반복적인 업데이트가, 적은 계산 노력으로 행해질 수 있다.

[0013] 일 바람직한 실시예에서, 수치 표현은 단일의 수치적 현재 콘텍스트 값의 이진 수치 표현이다. 바람직하게는, 이진 수치 표현의 비트들의 제1 서브셋(subset)은 하나 이상의 이전에 디코딩된 스펙트럼 값들과 연관된 제1 콘텍스트 서브구역 값에 의해 결정되고, 이진 수치 표현의 비트들의 제2 서브셋은 하나 이상의 이전에 디코딩된 스펙트럼 값들과 연관된 제2 콘텍스트 서브구역 값에 의해 결정되는데, 여기서 비트들의 제1 서브셋의 비트들은 비트들의 제2 서브셋의 비트들과는 다른 수치적 가중치를 포함한다. 그러한 표현은 수치적 이전 콘텍스트 값으로부터 수치적 현재 콘텍스트 값의 반복적인 도출에 매우 적합한 것으로 확인됐다.

[0014] 일 바람직한 실시예에서, 산술 디코더는, 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 수치적 이전 콘텍스트 값의 도출을 위해 고려되지 않은 콘텍스트 서브구역 값에 따라, 수치적 이전 콘텍스트 값의 수치 표현의 정보 비트들, 또는 수치적 이전 콘텍스트 값의 수치 표현의 비트 이동된 버전의 비트 방식 마스킹된(bit-wise masked) 서브셋을 수정하도록 구성된다. 수치적 이전 콘텍스트 값의 수치 표현에 대한 비트 방식 마스킹을 수행함으로써, 또는 수치적 이전 콘텍스트 값의 수치 표현을 비트 이동시킴으로써, 더 이상 이전처럼 상관없는 콘텍스트의 부분들이, 수치적 현재 값으로부터 제거되고, 바람직하게는, 현재 콘텍스트에 더 관련 있는 콘텍스트의 다른 부분들에 의해 대체되는 것이 이루어질 수 있다. 수치적 이전 콘텍스트 값의 수치 표현의 정보 비트들의 서브셋에 대한 비트 방식 마스킹은 콘텍스트 서브구역 값에 따라 수치적 이전 콘텍스트 값의 부분을 대체하는 것을 가능하게 하는데, 이는, 결국, 이전에 아직 고려되지 않은 콘텍스트의 부분을 고려하는 것을 가능하게 한다. 또한, 이동 연산은 이전 콘텍스트(즉, 스펙트럼 값들의 이전 튜플을 디코딩하는데 이용된 콘텍스트)를 결정하기 위해 이용된 이전에 디코딩된 스펙트럼 값들과 현재 콘텍스트(즉, 현재 디코딩되는 스펙트럼 값들의 디코딩을 위한 콘텍스트)를 결정하기 위해 이용된 이전에 디코딩된 스펙트럼 값들 사이에 약간의 중첩이 있다는 사실을 반영한다. 또한, 이동 연산들은 수치적 이전 콘텍스트 값을 이용하여 디코딩된 스펙트럼 값들에 대한 이전에 디코딩된 스펙트럼 값들의 주파수 연관성(예를 들어, 주파수에서 같음, 일 주파수 빈 만큼 주파수에서 더 큼, 등)은 수치적 현재 콘텍스트 값을 이용하여 디코딩되는 스펙트럼 값들에 대한 이전에 디코딩된 스펙트럼 값들의 주파수 관계와 다르다는 사실을 또한 반영한다.

[0015] 일 바람직한 실시예에서, 산술 디코더는, 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 각각 다른 콘텍스트 서브구역 값들과 연관된 비트들의 서브셋들의 수치적 가중치들이 수정되도록, 수치적 이전 콘텍스트 값의 수치 표현을 비트 이동하기 위해 구성된다. 이에 따라, 수치적 이전 콘텍스트 값을 이용하여 디코딩된 하나 이상의 스펙트럼 값들과 수치적 현재 콘텍스트 값을 이용하여 디코딩된 하나 이상의 스펙트럼 값들 사이의 주파수 위치의 이동이 효율적인 방식으로 수치적 현재 콘텍스트 값에 반영될 수 있다. 또한, 이동 연산은 일반적으로 표준 마이크로프로세서를 이용하여 적은 계산 노력으로 수행될 수 있다.

[0016] 일 바람직한 실시예에서, 산술 디코더는, 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 콘텍스트 서브구역 값과 연관되는 비트들의 서브셋이 수치 표현으로부터 삭제되도록, 수치적 이전 콘텍스트 값의 수치 표현을 비트 이동하기 위해 구성된다. 이에 따라, 단일 이동 연상에 의해, 두 가지 기능, 즉, 주파수 위치의 변경 고려, 및 수치적 이전 콘텍스트 값을 획득하기 위해 이용된 (콘텍스트 서브구역 값에 의해 표현된) 몇몇 스펙트럼 값들이 수치적 현재 콘텍스트 값을 획득하는데 더 이상 필요하지 않다는 사실에 대한 고려가 제공될 수

있다.

[0017] 일 바람직한 실시예에서, 산술 디코더는, (수치적 이전 콘텍스트 값을 이용하여 디코딩된) 이전에 디코딩된 스펙트럼 값들의 디코딩을 위해 고려되고 수치적 현재 콘텍스트 값을 이용하여 디코딩되는 스펙트럼 값들의 디코딩을 위해서는 고려되지 않는 콘텍스트 서브구역들과 연관된 비트들의 하나 이상의 서브셋들을 선택적으로 수정함으로써 수치적 이전 콘텍스트 값의 이진 수치 표현으로부터 수치적 현재 콘텍스트 값의 이진 수치 표현을 도출하기 위해, 콘텍스트 서브구역 값에 따라, 수치적 이전 콘텍스트 값의 이진 수치 표현의 비트들, 또는 수치적 이전 콘텍스트 값의 이진 수치 표현의 비트 이동된 버전의 제1 서브셋을 수정하고, 수치적 이전 콘텍스트 값의 이진 수치 표현의 비트들, 또는 수치적 이전 콘텍스트 값의 이진 수치 표현의 비트 이동된 버전의 제2 서브셋들을, 변경되지 않은 채로 두도록 구성된다. 이러한 구상은 특히 효율적인 것으로 증명되었다.

[0018] 일 바람직한 실시예에서, 산술 디코더는, 수치적 현재 콘텍스트 값의 수치 표현의 최하위 비트들의 서브셋이, 콘텍스트 상태가 수치적 현재 콘텍스트 값에 의해 정의되는 스펙트럼 값들의 디코딩을 위해 이용되고, 콘텍스트 상태가 수치적 다음(subsequent) 콘텍스트 값(예를 들어, 수치적 현재 콘텍스트 값으로부터 도출된 수치적 콘텍스트 값)에 의해 정의되는 스펙트럼 값들의 디코딩을 위해서는 이용되지 않는, 콘텍스트 서브구역 값을 기술하도록, 수치적 현재 콘텍스트 값의 수치 표현을 제공하기 위해 구성된다. 이러한 접근법은, 수치 표현의 최하위 비트들이 쉽게 이동될 수 있기 때문에, 이동 연산을 이용하여 수치적 이전 콘텍스트 값으로부터 수치적 현재 콘테스트 값을 도출(그리고 수치적 현재 콘텍스트 값으로부터 수치적 다음 콘텍스트 값을 도출)하는 것을 가능하게 한다. 또한, 수치적 이전 콘텍스트 값에 대해 있으나, 수치적 현재 콘텍스트 값에 대해서는 더 이상 관련 없는(또는, 동등하게, 수치적 현재 콘텍스트 값에 대해 관련 있으나, 수치적 다음 콘텍스트 값에 대해서는 더 이상 관련 없는) 그러한 콘텍스트 서브구역 값들에 작은 수치적 가중치를 할당하는 것이 적절한 것으로 또한 확인되었는데, 맵핑 규칙 인덱스 값으로의 수치적 (현재) 콘텍스트 값의 효율적인 맵핑을 가능하게 하기 때문이다.

[0019] 일 바람직한 실시예에서, 산술 디코더는, 수치적 현재 콘텍스트 값이 테이블의 엔트리에 의해 기술된 테이블 콘텍스트 값(예를 들어, 유효 상태 값)과 동일한지 또는 테이블의 엔트리들에 의해 기술된 구간 내에 있는지 여부를 결정하기 위해 적어도 하나의 테이블을 평가하고, 적어도 하나의 테이블에 대한 평가 결과에 따라 선택된 맵핑 규칙을 기술하는 맵핑 규칙 인덱스 값을 도출하도록 구성된다. 상기에서 기술된 바와 같이 구성되고 업데이트 되는 수치적 (현재) 콘텍스트 값은 맵핑 규칙 인덱스 값으로의 그러한 맵핑에 매우 적합한 것으로 확인됐다.

[0020] 일 바람직한 실시예에서, 산술 디코더는 복수의 콘텍스트 서브구역 값들의 합계가 미리 결정된 합계 임계 값보다 더 작은지 또는 미리 결정된 합계 임계 값과 같은지 여부를 검사하고, 검사 결과에 따라 수치적 현재 콘텍스트 값을 선택적으로 수정하도록 구성된다. 수치적 현재 콘텍스트 값의 그러한 추가의 선택적 수정은 수치적 콘텍스트 값의 업데이트를 위한 구상에 대해 어떠한 충돌도 없이 수치적 현재 콘텍스트 값 안에 의미 있는 콘텍스트 정보를 효율적으로 넣는데 매우 적합한 것으로 확인됐다.

[0021] 일 바람직한 실시예에서, 산술 디코더는, 콘텍스트 상태를 이용하여 디코딩되는 하나 이상의 스펙트럼 값들이 수치적 현재 콘텍스트 값에 정의됨으로써 오디오 콘텐츠의 동일한 시간 부분과 연관되고, 수치적 현재 콘텍스트 값에 의해 정의된 콘텍스트 상태를 이용하여 디코딩되는 하나 이상의 스펙트럼 값들보다 더 낮은 주파수들과 연관되는, 복수의 콘텍스트 서브구역 값들의 합계가 미리 결정된 합계 임계 값보다 더 작은지 또는 미리 결정된 합계 임계 값과 같은지 여부를 검사하고, 검사 결과에 따라 수치적 현재 콘텍스트 값을 선택적으로 수정하도록 구성된다. 비교적 작은 스펙트럼 값들의 구역의 존재를 식별하기 위한 그러한 검사는 가치 있는 추가 정보를 제공하는 것으로 확인됐다.

[0022] 일 바람직한 실시예에서, 산술 디코더는, 제1 복수의 이전에 디코딩된 스펙트럼 값들과 연관된 제1 콘텍스트 서브구역 값을 획득하기 위해 제1 복수의 이전에 디코딩된 스펙트럼 값들의 절대 값들을 합계하고, 제2 복수의 이전에 디코딩된 스펙트럼 값들과 연관된 제2 콘텍스트 서브구역 값을 획득하기 위해 제2 복수의 이전에 디코딩된

스펙트럼 값들의 절대 값들 합계하도록 구성된다. 이에 따라, 각각 다른 콘텍스트 서브구역 값들이 획득될 수 있다.

[0023] 일 바람직한 실시예에서, 산술 디코더는, 수치적 이전 콘텍스트 값의 수치 표현의 정보 비트들의 트루(true) 서브셋을 이용하여 콘텍스트 서브구역 값들이 표현될 수 있도록, 콘텍스트 서브구역 값들을 제한하기 위해 구성된다. 콘텍스트 서브구역 값들의 제한은 콘텍스트 서브구역 값들의 정보 콘텐츠에 상당한 해로운 영향을 미치지 않는 것으로 확인됐다. 그러나, 그러한 제한은 콘텍스트 서브구역 값을 표현하기 위해 요구되는 비트들의 수를 상당히 작게 유지되게 할 수 있다는 이점을 가져오는데, 이는 메모리 요구에 긍정적인 영향을 준다. 또한, 콘텍스트 서브구역 값들의 제한은 수치적 현재 콘텍스트 값의 반복적인 업데이트를 용이하게 한다.

[0024] 본 발명의 다른 실시예는 입력된 오디오 정보에 기초하여 인코딩된 오디오 정보를 제공하기 위한 오디오 인코더를 고안한다. 오디오 인코더는 입력된 오디오 정보의 시간 도메인 표현에 기초하여 주파수 도메인 오디오 표현을 제공하기 위한 에너지 압축 시간 도메인 대 주파수 도메인 변환기를 포함하여, 주파수 도메인 오디오 표현이 스펙트럼 값들의 셋트를 포함한다. 오디오 인코더는 또한, 가변 길이 코드워드를 이용하여, 스펙트럼 값, 또는 그의 전처리된(preprocessed) 버전, 또는 - 동등하게 - 복수의 스펙트럼 값들 그의 전처리된 버전을 인코딩하도록 구성되는 산술 인코더를 포함한다. 산술 인코더는 스펙트럼 값, 또는 스펙트럼 값의 최상위 비트 평면의 값, 또는 스펙트럼 값의 최상위 비트 평면의 값을, 코드 값에 맵핑하도록 구성된다. 산술 인코더는, 수치적 현재 콘텍스트 값에 의해 기술된 콘텍스트 상태에 따라 코드 값으로의, 스펙트럼 값, 또는 스펙트럼 값의 최상위 비트 평면의 값의 맵핑을 기술하는 맵핑 규칙을 선택하도록 구성된다. 산술 인코더는 복수의 이전에 인코딩된 스펙트럼 값들에 따라 수치적 현재 콘텍스트 값을 결정하도록 구성된다. 산술 인코더는, 디코딩되는 상기 하나 이상의 스펙트럼 값들과 연관된 콘텍스트 상태를 기술하는(또는, 좀더 정확히, 인코딩되는 상기 하나 이상의 스펙트럼 값들의 인코딩을 위한 콘텍스트 상태를 기술하는) 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 콘텍스트 서브구역 값에 따라, 하나 이상의 이전에 인코딩된 스펙트럼 값들과 연관된 콘텍스트 상태를 기술하는(또는, 좀더 정확히, 상기 하나 이상의 이전에 인코딩된 스펙트럼 값들의 인코딩을 위한 콘텍스트 상태를 기술하는), 수치적 이전의 콘텍스트 값의 수치 표현을 수정하도록 구성된다.

[0025] 상기 오디오 인코더는 상기 오디오 디코더와 동일한 조사결과들(findings)에 기초한다. 또한, 상기 오디오 인코더는 상기 오디오 디코더에 대해 논의된 기능들에 의해 보완될 수 있다.

[0026] 본 발명에 따른 다른 실시예는 인코딩된 오디오 정보에 기초하여 디코딩된 오디오 정보를 제공하기 위한 방법을 고안한다.

[0027] 본 발명에 따른 다른 실시예는 입력된 오디오 정보에 기초하여 인코딩된 오디오 정보를 제공하기 위한 방법을 고안한다.

[0028] 본 발명에 따른 다른 실시예는 상기 방법들 중 하나를 수행하기 위한 컴퓨터 프로그램을 고안한다.

도면의 간단한 설명

[0029] 본 발명에 따른 실시예들이 첨부된 도면들을 참조하여 이어서 기술될 것인데:

도 1은 본 발명의 일 실시예에 따른 오디오 인코더의 블록 도식도;

도 2는 본 발명의 일 실시예에 따른 오디오 디코더의 블록 도식도;

도 3은 스펙트럼 값들을 디코딩하기 위한 알고리즘 "values_decode()"의 의사(pseudo) 프로그램 코드 표현을 도시하는 도면;

도 4는 상태 계산을 위한 콘텍스트의 도식적 표현을 도시하는 도면;

도 5a는 콘텍스트를 맵핑하기 위한 알고리즘 "arith_map_context()"의 의사 프로그램 코드 표현을 도시하는 도면;

도 5b는 콘텍스트를 맵핑하기 위한 다른 알고리즘 "arith_map_context()"의 의사 프로그램 코드 표현을 도시하는 도면;

도 5c는 콘텍스트 상태 값을 획득하기 위한 알고리즘 "arith_get_context()"의 의사 프로그램 코드 표현을 도시하는 도면;

도 5d는 콘텍스트 상태 값을 획득하기 위한 다른 알고리즘 "arith_get_context()"의 의사 프로그램 코드 표현을 도시하는 도면;

도 5e는 상태 값(또는 상태 변수)으로부터 누적 빈도 테이블 인덱스 값 "pki"를 도출하기 위한 알고리즘 "arith_get_pk()"의 의사 프로그램 코드 표현을 도시하는 도면;

도 5f는 상태 값(또는 상태 변수)으로부터 누적 빈도 테이블 인덱스 값 "pki"를 도출하기 위한 다른 알고리즘 "arith_get_pk()"의 의사 프로그램 코드 표현을 도시하는 도면;

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

도 5h는 가변 길이 코드워드로부터 심볼을 산술적으로 디코딩하기 위한 다른 알고리즘 "arith_decode()"의 의사 프로그램 코드 표현의 제1 부분을 도시하는 도면;

도 5i는 가변 길이 코드워드로부터 심볼을 산술적으로 디코딩하기 위한 다른 알고리즘 "arith_decode()"의 의사 프로그램 코드 표현의 제2 부분을 도시하는 도면;

도 5j는 공통 값 m으로부터 스펙트럼 값들의 절대 값들 a, b를 도출하기 위한 알고리즘의 의사 프로그램 코드 표현을 도시하는 도면;

도 5k는 디코딩된 스펙트럼 값들의 어레이로 디코딩된 값들 a, b를 입력하기 위한 알고리즘의 의사 프로그램 코드 표현을 도시하는 도면;

도 5l은 디코딩된 스펙트럼 값들의 절대 값들 a, b에 기초하여 콘텍스트 서브구역(subregion) 값을 획득하기 위한 알고리즘 "arith_update_context()"의 의사 프로그램 코드 표현을 도시하는 도면;

도 5m은 디코딩된 스펙트럼 값들의 어레이와 콘텍스트 서브구역 값들의 어레이의 엔트리들을 채우기 위한 알고리즘 "arith_finish()"의 의사 프로그램 코드 표현을 도시하는 도면;

도 5n은 공통 값 m으로부터 디코딩된 스펙트럼 값들의 절대 값들 a, b를 도출하기 위한 다른 알고리즘의 의사 프로그램 코드 표현을 도시하는 도면;

도 5o는 디코딩된 스펙트럼 값들의 어레이 및 콘텍스트 서브구역 값들의 어레이를 업데이트하기 위한 알고리즘 "arith_update_context()"의 의사 프로그램 코드 표현을 도시하는 도면;

도 5p는 디코딩된 스펙트럼 값들의 어레이의 엔트리들 및 콘텍스트 서브구역 값들의 어레이의 엔트리들을 채우기 위한 알고리즘 "arith_save_context()"의 의사 프로그램 코드 표현을 도시하는 도면;

도 5q는 정의에 대한 범례를 도시하는 도면;

도 5r은 정의에 대한 다른 범례를 도시하는 도면;

도 6a는 통합 음성 오디오 코딩(USAC) 미가공 데이터 블록에 대한 구문 표현을 도시하는 도면;

도 6b는 단일 채널 성분에 대한 구문 표현을 도시하는 도면;

도 6c는 채널 쌍 성분에 대한 구문 표현을 도시하는 도면;

도 6d는 "ICS" 제어 정보에 대한 구문 표현을 도시하는 도면;

도 6e는 주파수 도메인 채널 스트림에 대한 구문 표현을 도시하는 도면;

도 6f는 산술적으로 코딩된 스펙트럼 데이터에 대한 구문 표현을 도시하는 도면;

도 6g는 스펙트럼 값들의 셋트를 디코딩하기 위한 구문 표현을 도시하는 도면;

도 6h는 스펙트럼 값들의 셋트를 디코딩하기 위한 다른 구문 표현을 도시하는 도면;

도 6i는 데이터 성분들 및 변수들에 대한 범례를 도시하는 도면;

도 6j는 데이터 성분들 및 변수들에 대한 다른 범례를 도시하는 도면;

도 7은 본 발명의 제1 양상에 따른 오디오 인코더에 대한 블록 도식도;

도 8은 본 발명의 제1 양상에 따른 오디오 디코더에 대한 블록 도식도;

도 9는 본 발명의 제1 양상에 따른 맵핑 규칙 인덱스 값으로의 수치적 현재 콘텍스트 값의 맵핑에 대한 그래프 표현을 도시하는 도면;

도 10은 본 발명의 제2 양상에 따른 오디오 인코더에 대한 블록 도식도;

도 11은 본 발명의 제2 양상에 따른 오디오 디코더에 대한 블록 도식도;

도 12는 본 발명의 제3 양상에 따른 오디오 인코더에 대한 블록 도식도;

도 13은 본 발명의 제3 양상에 따른 오디오 디코더에 대한 블록 도식도;

도 14a는, USAC 표준 초안(USAC Draft Standard)의 규격 초안 4에 따라 이용되는 상태 계산을 위한 콘텍스트에 대한 도식적 표현을 도시하는 도면;

도 14b는 USA 표준 초안의 규격 초안 4에 따른 산술 코딩 기법에서 이용된 테이블들에 대한 개관을 도시하는 도면;

도 15a는, 본 발명에 따른 실시예들에서 이용되는 상태 계산을 위한 콘텍스트에 대한 도식적 표현을 도시하는 도면;

도 15b는 본 발명에 따른 산술적 코딩 기법에서 이용된 테이블들에 대한 개관을 도시하는 도면;

도 16a는 본 발명, 및 USAC 표준 초안의 규격 초안 5, 및 AAC(고급 오디오 코딩, advanced audio coding) 허프만(Huffman) 코딩에 따른 무잡음 코딩 기법에 대한 읽기 전용 메모리 요구의 그래프 표현을 도시하는 도면;

도 16b는 본 발명 및 USAC 표준 초안의 규격 초안 5에 따르는 구상에 따른 전체 USAC 디코더 데이터 읽기 전용 메모리 요구에 대한 그래프 표현을 도시하는 도면;

도 17은 USAC 표준 초안의 규격 초안 3 또는 규격 초안 5에 따른 무잡음 코딩과 본 발명에 따른 코딩 기법의 비교를 위한 배치에 대한 도식적 표현을 도시하는 도면;

도 18은 USAC 표준 초안의 규격 초안 3 및 본 발명의 일 실시예에 따르는 USAC 산술 코더에 의해 산출되는 평균 비트율에 대한 테이블 표현을 도시하는 도면;

도 19는 USAC 표준 초안의 규격 초안 3에 따른 산술 디코더와 본 발명의 일 실시예에 따른 산술 디코더에 대한 최소 및 최대 비트 보유 레벨에 대한 테이블 표현을 도시하는 도면;

도 20은 산술 디코더의 각각 다른 버전들에 대하여 USAC 표준 초안의 규격 초안 3에 따라 32 kbits 스트림을 디코딩하기 위한 평균 복잡도 수치들에 대한 테이블 표현을 도시하는 도면;

도 21a 및 21b는 테이블 "ari_lookup_m[600]"의 콘텐츠에 대한 테이블 표현을 도시하는 도면;

도 22a 내지 22d는 테이블 "ari_hash_m[600]"의 콘텐츠에 대한 테이블 표현을 도시하는 도면;

도 23a 내지 23h은 테이블 "ari_cf_m[96][17]"의 콘텐츠에 대한 테이블 표현을 도시하는 도면; 및

도 24는 테이블 "ari_cf_r[]"의 콘텐츠에 대한 테이블 표현을 도시하는 도면.

발명을 실시하기 위한 구체적인 내용

[0030] 1. 도 7에 따른 오디오 인코더

[0031] 도 7은 본 발명의 일 실시예에 따른 오디오 인코더에 대한 블록 도식도를 도시한다. 오디오 인코더(700)는 입력된 오디오 정보(710)를 수신하고, 그에 기초하여, 인코딩된 오디오 정보(712)를 제공하도록 구성된다. 오디오 인코더는 입력된 오디오 신호(710)의 시간 도메인 표현에 기초하여 주파수 도메인 오디오 표현(722)을 제공하도록 구성되는 에너지 압축 시간 도메인 대 주파수 도메인 변환기(720)을 포함하여, 주파수 도메인 오디오 표현(722)아 스펙트럼 값들의 셋트를 포함한다. 오디오 인코더(700)는 또한, (예를 들어, 복수의 가변 길이 코드워드들을 포함할 수 있는) 인코딩된 오디오 정보(712)를 획득하기 위해 가변 길이 코드워드를 이용하여, (주파수 도메인 오디오 표현(722)을 형성하는 스펙트럼 값들의 셋트 중에서) 하나의 스펙트럼 값, 또는 그의 전처리된 버전을 인코딩하도록 구성되는 산술 인코더(730)를 포함한다.

[0032] 산술 인코더(730)는 콘텍스트 상태에 따라 코드 값(즉, 가변 길이 코드워드)으로, 스펙트럼 값, 또는 스펙트럼 값의 최상위 비트 평면의 값을 맵핑하도록 구성된다. 산술 인코더는 (현재) 콘텍스트 상태에 따라, 코드 값으로의, 스펙트럼 값, 또는 스펙트럼 값의 최상위 비트 평면의 맵핑을 기술하는 맵핑 규칙을 선택하도록 구성된다. 산술 인코더는, 복수의 이전에 인코딩된 (바람직하게는, 그러나 반드시 그렇지는 않는, 인접한) 스펙트럼 값들에 따라, 현재 콘텍스트 상태, 또는 현재 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 상태 값을 결정하도록 구성된다. 이를 위해, 산술 인코더는 그 엔트리들이 수치적 콘텍스트 값들 중에서 유효 상태 값들과 수치적 콘텍스트 값들에 대한 구간들의 경계들을 모두 정의하는 해시 테이블을 평가하도록 구성되는데, 여기서 맵핑 규칙 인덱스 값은 유효 상태 값인 수치적 (현재) 콘텍스트 값에 개별적으로 연관되고, 여기서 공통 맵핑 규칙 인덱스 값은 구간 경계들(여기서 구간 경계들은 바람직하게는 해시 테이블의 엔트리들에 의해 정의된다)에 의해 경계지어진 구간 내에 있는 각각 다른 수치적 (현재) 콘텍스트 값들에 연관된다.

[0033] 알 수 있는 바와 같이, (인코딩된 오디오 정보(712)의) 코드 값으로의, (주파수 도메인 오디오 표현(722)의) 스펙트럼 값, 또는 스펙트럼 값이 최상위 비트 평면의 맵핑이, 맵핑 규칙(742)을 이용하여 스펙트럼 값 인코딩(740)에 의해 수행될 수 있다. 상태 추적기(state tracker, 750)는 콘텍스트 상태를 추적하도록 구성될 수 있다. 상태 추적기(750)는 현재 콘텍스트 상태를 기술하는 정보(754)를 제공한다. 현재 콘텍스트 상태를 기술하는 정보(754)는 바람직하게는 수치적 현재 콘텍스트 값의 형태를 취할 수 있다. 맵핑 규칙 선택기(760)는, 코드 값으로의, 스펙트럼 값, 또는 스펙트럼 값의 최상위 비트 평면의 맵핑을 기술하는 맵핑 규칙, 예를 들어, 누적 빈도 테이블을 선택하도록 구성된다. 이에 따라, 맵핑 규칙 선택기(760)는 스펙트럼 값 인코딩(740)에 맵핑 규칙 정보(742)를 제공한다. 맵핑 규칙 정보(742)는 맵핑 규칙 인덱스 값 또는 맵핑 규칙 인덱스 값에 따라 선택된 누적 빈도 테이블의 형태를 취할 수 있다. 맵핑 규칙 선택기(760)는 그 엔트리들이 수치적 콘텍스트 값들 중에서 유효 상태 값들과 수치적 콘텍스트 값들의 구간들의 경계들을 모두 정의하는 해시 테이블(752)을 포함하는데(또는 적어도 평가하는데), 여기서 맵핑 규칙 인덱스 값은 유효 상태 값인 수치적 콘텍스트 값에 개별적으로 연관되고, 여기서 공통 맵핑 규칙 인덱스 값은 구간 경계들에 의해 경계지어진 구간 내에 있는 각각 다른 수치적 콘텍스트 값들에 연관된다. 해시 테이블(762)은 맵핑 규칙을 선택하기 위해, 즉, 맵핑 규칙 정보(742)를 제공하기 위해 평가된다.

[0034] 상기를 요약하면, 오디오 인코더(700)는 시간 도메인 대 주파수 도메인 변환기에 의해 제공된 주파수 도메인 오디오 표현의 산술 인코딩을 수행한다. 산술 인코딩이 콘텍스트에 따르므로(contenxt-dependent), 맵핑 규칙(즉, 누적 빈도 테이블)이 이전에 인코딩된 스펙트럼 값들에 따라 선택된다. 이에 따라, 서로 및/또는 현재 인코딩된 스펙트럼 값(즉, 현재 인코딩된 스펙트럼 값의 미리 결정된 환경 내의 스펙트럼 값들)에 시간 및/또는 주파수(또는, 적어도, 미리 결정된 환경 내에서)에서 인접한 스펙트럼 값들은 산술 인코딩에 의해 평가된 확률 분포를 조절하기 위해 산술 인코딩에서 고려된다. 적합한 맵핑 규칙을 선택할 때, 상태 추적기(750)에 의해 제공된 수치적 현재 콘텍스트 값들(754)이 평가된다. 일반적으로 각각 다른 맵핑 규칙들의 수가 수치적 현재 콘텍스트 값들(754)의 가능한 값들의 수보다 상당히 작기 때문에, 맵핑 규칙 선택기(760)는 비교적 많은 수의 각각 다른 수치적 콘텍스트 값들에 (예를 들어, 맵핑 규칙 인덱스 값에 의해 기술된) 동일한 맵핑 규칙을 할당한다. 그럼에도 불구하고, 일반적으로, 좋은 코딩 효율성을 획득하기 위해 특정한 맵핑 규칙에 연관되어야 하는 (특정 수치

적 콘텍스트 값들에 의해 표현된) 특정 스펙트럼 구성들이 있다.

[0035] 만약 단일 해시 테이블의 엔트리들이 수치적 (현재) 콘텍스트 값들의 유효 상태 값들과 구간들의 경계들을 모두 정의한다면, 수치적 현재 콘텍스트 값에 따르는 맵핑 규칙의 선택이 특히 높은 계산 효율성을 가지며 수행될 수 있는 것으로 확인됐다. 이러한 매커니즘은 맵핑 규칙 선택의 요구에 잘 적응되는 것으로 확인됐는데, 이는 단일 유효 상태 값(또는 유효 수치적 콘텍스트 값)이 (공통 맵핑 규칙이 연관되는) 복수의 비 유효(non-significant) 상태 값들의 왼쪽 구간과 (공통 맵핑 규칙이 연관되는) 복수의 비 유효 상태 값들의 오른쪽 구간 사이에 끼워지는 많은 경우들이 있기 때문이다. 또한, 그 엔트리들이 수치적 (현재) 콘텍스트값들의 유효 상태 값들과 구간들의 경계들을 모두 정의하는 단일 해시 테이블을 이용하는 매커니즘은, 예를 들어, 그 사이에 유효 상태 값이 없이 (비 유효 수치적 콘텍스트 값들이라고도 지칭된) 비 유효 상태 값들의 두 개의 인접한 구간들이 있는 각각 다른 경우들을 효율적으로 다룰 수 있다. 테이블 접근 횟수가 적게 유지됨으로 인해 특히 높은 계산 효율성이 달성된다. 예를 들어, 수치적 현재 콘텍스트 값이 임의의 유효 상태 값들과 같은지 여부, 또는 비 유효 상태 값들의 구간들 중 어느 것에 수치적 현재 콘텍스트 값이 있는지 여부를 알아내기 위해 대부분의 실시예들에서 단일 반복 테이블 검사로 충분하다. 결과적으로, 시간과 에너지를 모두 소비하는 테이블 접근 횟수가 작게 유지될 수 있다. 그러므로, 해시 테이블(762)을 이용하는 맵핑 규칙 선택기(760)는 계산 복잡도 측면에서 특히 효율적인 맵핑 규칙 선택기로 여겨질 수 있으며, 한편 여전히 (비트율 측면에서) 좋은 인코딩 효율성을 획득하는 것을 가능하게 한다.

[0036] 수치적 현재 콘텍스트 값(754)으로부터의 맵핑 규칙 정보(742)의 도출에 관한 더욱 세부적인 사항들이 하기에서 기술될 것이다.

[0037] 2. 도 8에 따른 오디오 디코더

[0038] 도 8은 오디오 디코더(800)에 대한 블록 도식도를 도시한다. 오디오 디코더(800)는 인코딩된 오디오 신호(810)를 수신하고, 그에 기초하여, 디코딩된 오디오 신호(812)를 제공하도록 구성된다. 오디오 디코더(800)는 스펙트럼 값들의 산술적으로 인코딩된 표현(821)에 기초하여 복수의 스펙트럼 값들(822)을 제공하도록 구성되는 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 도메인 오디오 표현(812)을 제공하도록 구성되는 주파수 도메인 대 시간 도메인 변환기(830)를 포함한다.

[0039] 산술 디코더(820)는 하나 이상의 디코딩된 스펙트럼 값들, 또는 하나 이상의 디코딩된 스펙트럼 값들의 적어도 일부분(예를 들어, 최상위 비트 평면)을 표현하는 심볼 코드에 스펙트럼 값들의 산술적으로 인코딩된 표현(821)의 코드 값을 맵핑하도록 구성되는 스펙트럼 값 결정기(824)를 포함한다. 스펙트럼 값 결정기(824)는 맵핑 규칙 정보(828a)에 의해 기술될 수 있는 맵핑 규칙에 따라 맵핑을 수행하도록 구성될 수 있다. 맵핑 규칙 정보(828a)는, 예를 들어, 맵핑 규칙 인덱스 값, 또는 (예를 들어, 맵핑 규칙 인덱스 값에 따라 선택된) 선택된 누적 빈도 테이블의 형태를 취할 수 있다.

[0040] 산술 디코더(820)는 (콘텍스트 상태 정보(826a)에 의해 기술될 수 있는) 콘텍스트 상태에 따라 (하나 이상의 스펙트럼 값들, 또는 그의 최상위 비트 평면을 기술하는) 심볼 코드로의 (스펙트럼 값들의 산술적으로 인코딩된 표현(821)에 의해 기술된) 코드 값들의 맵핑을 기술하는 맵핑 규칙(예를 들어, 누적 빈도 테이블)을 선택하도록 구성된다. 산술 디코더(820)는 복수의 이전에 디코딩된 스펙트럼 값들에 따라 (수치적 현재 콘텍스트 값에 의해 기술된) 현재 콘텍스트 상태를 결정하도록 구성된다. 이를 위해, 이전에 디코딩된 스펙트럼 값들을 기술하는 정보를 수신하고, 그에 기초하여, 현재 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값(826a)을 제공하는 상태 추적기(826)가 이용될 수 있다.

[0041] 산술 디코더는 또한, 맵핑 규칙을 선택하기 위해, 그 엔트리들이 수치적 콘텍스트 값들 중에서 유효 상태 값들과 수치적 콘텍스트 값들의 구간들의 경계들을 모두 정의하는 해시 테이블(829)를 평가하도록 구성될 수 있는데, 여기서, 맵핑 규칙 인덱스 값은 유효 상태 값인 수치적 콘텍스트 값에 개별적으로 연관되고, 여기서 공통 맵핑 규칙 인덱스 값은 구간 경계들에 의해 경계지어진 구간 내에 있는 각각 다른 수치적 콘텍스트 값들에 연관된다. 해시 테이블(829)의 평가는, 예를 들어, 맵핑 규칙 선택기(828)의 부분일 수 있는 해시 테이블 평가기를 이용하여 수행될 수 있다. 이에 따라, 맵핑 규칙 정보(828a)는, 예를 들어, 맵핑 규칙 인덱스 값의 형태로, 현재 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값(826a)에 기초하여 획득된다. 맵핑 규칙 선택기(828)는, 예를 들어, 해시 테이블(829)의 평가 결과에 따라 맵핑 규칙 인덱스 값(828a)을 결정할 수 있다. 그렇지 않으면, 해시 테이블(829)의 평가는 맵핑 규칙 인덱스 값을 바로 제공할 수 있다.

[0042] 오디오 신호 디코더(800)의 기능과 관련하여, 산술 디코더(820)는, 대체로, 디코딩되는 스펙트럼 값들에 잘 적응되는 맵핑 규칙(예를 들어, 누적 빈도 테이블)을 선택하도록 구성되는데, 이는 맵핑 규칙이 (예를 들어, 수치적 현재 콘텍스트 값에 의해 기술된) 현재 콘텍스트 상태에 따라 선택되기 때문이며, 이는 결국 복수의 이전 디코딩된 스펙트럼 값들에 따라 결정된다는 것을 알아야 한다. 이에 따라, 디코딩되는 인접한 스펙트럼 값들 사이의 통계적 의존성이 활용될 수 있다. 또한, 산술 디코더(820)는, 맵핑 규칙 선택기(828)를 이용하여, 계산 복잡도, 테이블 크기, 및 코딩 효율성 사이의 좋은 균형을 지니며 효율적으로 구현될 수 있다. 그 엔트리들이 유효 상태 값들과 비 유효 상태 값들의 구간들의 구간 경계들을 기술하는 (단일) 해시 테이블(829)을 평가하여, 수치적 현재 콘텍스트 값(826a)으로부터 맵핑 규칙 정보(828a)를 도출하기 위해 단회(single) 반복 테이블 검색만으로 충분하다. 이에 따라, 비교적 적은 수의 각각 다른 맵핑 규칙 인덱스 값들에 비교적 많은 수의 각각 다른 가능한 수치적 (현재) 콘텍스트 값들을 맵핑하는 것이 가능하다. 해시 테이블(829)을 이용하여, 상기에서 기술한 바와 같이, 많은 경우에, 비 유효 상태 값들(비 유효 콘텍스트 값들)의 왼쪽 구간과 비 유효 상태 값들(비 유효 콘텍스트 값들)의 오른쪽 구간 사이에 단일의 분리된 유효 상태 값들(유효 콘텍스트 값)이 끼워 넣어진다는 결과를 활용하는 것이 가능한데, 여기서 왼쪽 구간의 상태 값들(콘텍스트 값들)과 오른쪽 구간의 상태 값들(콘텍스트 값들)을 비교했을 때, 각각 다른 맵핑 규칙 인덱스 값이 유효 상태 값(유효 콘텍스트 값)과 연관된다. 그러나, 해시 테이블(829)의 사용은 또한 그 사이에 유효 상태 값이 없이, 수치적 상태 값들의 두 개의 구간들이 바로 옆에 인접한 경우에 매우 적합하다.

[0043] 결론은, 해시 테이블(829)을 평가하는 맵핑 규칙 선택기(828)는 현재 콘텍스트 상태에 따라 (또는 현재 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값에 따라) 맵핑 규칙을 선택할 때 (또는 맵핑 규칙 인덱스 값을 제공할 때) 특히 좋은 효율성을 가져오는데, 이는 해싱 매커니즘이 오디오 디코더에서 일반적인 콘텍스트 시나리오들에 잘 적응되기 때문이다.

[0044] 더 자세한 사항들이 하기에서 기술될 것이다.

[0045] 3. 도 9에 따른 콘텍스트 값 해싱 매커니즘

[0046] 다음에서는, 맵핑 규칙 선택기(760) 및/또는 맵핑 규칙 선택기(828)에서 구현될 수 있는 콘텍스트 해싱 매커니즘이 기술될 것이다. 해시 테이블(762) 및/또는 해시 테이블(829)은 상기 콘텍스트 값 해싱 매커니즘을 구현하기 위해 이용될 수 있다.

[0047] 이제 수치적 현재 콘텍스트 값 해싱 시나리오를 도시하는 도 9를 참조하여, 더 자세한 사항들이 기술될 것이다. 도 9의 그래프 표현에서, 가로 좌표(910)는 수치적 현재 콘텍스트 값(즉, 수치적 콘텍스트 값들)의 값들을 기술한다. 세로 좌표(912)는 맵핑 규칙 인덱스 값들을 기술한다. 표시들(914)은 (비 유효 상태들을 기술하는) 비 유효 수치적 콘텍스트 값들에 대한 맵핑 규칙 인덱스 값들을 기술한다. 표시들(916)은 개개의 (참인(true)) 유효

상태들을 기술하는 "개개의" (참인) 유효 수치적 콘텍스트 값들에 대한 맵핑 규칙 인덱스 값들을 기술한다. 표시들(916)은 "부적절한" 유효 상태들을 기술하는 "부적절한" 수치적 콘텍스트 값들에 대한 맵핑 규칙 인덱스 값들을 기술하는데, 여기서 "부적절한" 유효 상태는 동일한 맵핑 규칙 인덱스 값이 비 유효 수치적 콘텍스트 값들의 인접한 구간들 중 하나에 대해 연관되는 유효 상태이다.

[0048] 알 수 있는 바와 같이, 해시 테이블 엔트리 "ari_hash_m[i1]"은 수치적 콘텍스트 값 c1을 갖는 개개의 (참인) 유효 상태를 기술한다. 알 수 있는 바와 같이, 맵핑 규칙 인덱스 값 mriv1은 수치적 콘텍스트 값 c1을 갖는 개개의 (참인) 유효 상태에 연관된다. 이에 따라, 수치적 콘텍스트 값 c1과 맵핑 규칙 인덱스 값 mriv1은 모두 해시 테이블 엔트리 "ari_hash_m[i1]"에 의해 기술된다. 수치적 콘텍스트 값들의 구간(932)은 수치적 콘텍스트 값 c1에 의해 경계지어지는데, 여기서 수치적 콘텍스트 값 c1은 구간 932에 속하지 않아, 구간 932의 가장 큰 수치적 콘텍스트 값은 c1-1과 같다. (mriv1과는 다른) 맵핑 규칙 인덱스 값 mriv4는 구간 932의 수치적 콘텍스트 값들과 연관된다. 맵핑 규칙 인덱스 값 mriv4는, 예를 들어, 추가 테이블 "ari_lookup_m"의 테이블 엔트리 "ari_lookup_m[i1-1]"에 의해 기술될 수 있다.

[0049] 또한, 맵핑 규칙 인덱스 값 mriv2는 구간 934 내에 있는 수치적 콘텍스트 값들과 연관될 수 있다. 구간 934의 하부 경계는 유효 수치적 콘텍스트 값인 수치적 콘텍스트값 c1에 의해 결정되는데, 여기서 수치적 콘텍스트 값 c1은 구간 932에 속하지 않는다. 이에 따라, 구간 934의 가장 작은 값은 (정수 수치적 콘텍스트 값들이라고 추정하면) c1+1과 같다. 구간 934의 다른 경계는 수치적 콘텍스트 값 c2에 의해 결정되는데, 여기서 수치적 콘텍스트 값 c2는 구간 934에 속하지 않아, 구간 934의 가장 큰 값이 c2-1과 같다. 수치적 콘텍스트 값 c2는 해시 테이블 엔트리 "ari_hash_m[i2]"에 의해 기술되는 이른바 "부적절한" 수치적 콘텍스트 값이다. 예를 들어, 맵핑 규칙 인덱스 값 mriv2가 수치적 콘텍스트 값 c2와 연관될 수 있어서, "부적절한" 유효 수치적 콘텍스트 값 c2와 연관된 수치적 콘텍스트 값이 수치적 콘텍스트 값 c2에 의해 경계지어진 구간 934와 연관된 맵핑 규칙 인덱스 값과 같다. 또한, 수치적 콘텍스트 값의 구간 936은 또한 수치적 콘텍스트 값 c2에 의해 경계지어지는데, 여기서 수치적 콘텍스트 값 c2가 구간 936에 속하지 않아, 구간 936의 가장 작은 수치적 콘텍스트 값은 c2+1과 같다. 일반적으로 맵핑 규칙 인덱스 값 mriv2와 다른 맵핑 규칙 인덱스 값 mriv3는 구간 936의 수치적 콘텍스트 값들과 연관된다.

[0050] 알 수 있는 바와 같이, 수치적 콘텍스트 값들의 구간 932에 연관되는 맵핑 규칙 인덱스 값 mriv4는 테이블 "ari_lookup_m"의 엔트리 "ari_lookup_m[i1-1]"에 의해 기술될 수 있으며, 구간 934의 수치적 콘텍스트 값들에 연관되는 맵핑 규칙 인덱스 값 mriv2는 테이블 "ari_lookup_m"의 엔트리 "ari_lookup_m[i1]"에 의해 기술될 수 있고, 맵핑 규칙 인덱스 값 mriv3는 테이블 "ari_lookup_m"의 엔트리 "ari_lookup_m[i2]"에 의해 기술될 수 있다. 여기서 주어진 예시에서, 해시 테이블 인덱스 값 i2는 해시 테이블 인덱스 값 i1보다 1 만큼 더 클 수 있다.

[0051] 도 9에서 알 수 있는 바와 같이, 맵핑 규칙 선택기 760 또는 맵핑 규칙 선택기 828은 수치적 현재 콘텍스트 값 764, 826a를 수신하고, ("개개의" 유효 상태 값인지 또는 "부적절한" 유효 상태 값인지 여부에 관계없이) 수치적 현재 콘텍스트 값이 유효 상태 값인지 여부, 또는 ("개개의" 또는 "부적절한") 유효 상태 값들 c1, c2에 의해 경계지어지는 구간들 932, 934, 936 중의 하나 안에 수치적 현재 콘텍스트 값이 있는지 여부를, 테이블 "ari_hash_m" 엔트리들을 평가하여 결정할 수 있다. 수치적 현재 콘텍스트 값이 유효 상태 값 c1, c2와 같은지 여부에 대한 검사 및 (수치적 현재 콘텍스트 값이 유효 상태 값과 같지 않은 경우에) 구간들 932, 934, 936 중의 어느 것에 수치적 현재 상태 값이 있는지에 대한 평가는 모두 단일의 공통 해시 테이블 검색을 이용하여 수행될 수 있다.

[0052] 또한, 해시 테이블 "ari_hash_m"의 평가는 해시 테이블 인덱스 값(예를 들어, i1-1, i1, 또는 i2)을 획득하기 위해 이용될 수 있다. 그러므로, 맵핑 규칙 선택기(760, 828)는, 단일 해시 테이블(762, 829)(예를 들어, 해시 테이블 "ari_hash_m")을 평가하여, 유효 상태 값(예를 들어, c1 또는 c2) 및/또는 구간(예를 들어, 932, 934, 936)을 지칭하는 해시 테이블 인덱스 값(예를 들어, i1-1, i1, 또는 i2) 및 수치적 현재 콘텍스트 값이 (유효

상태 값이라고도 지칭된) 유효 콘텍스트 값인지 아닌지 여부에 관한 정보를 얻도록 구성될 수 있다.

[0053] 또한, 해시 테이블(762, 829, "ari_hash_m")의 평가에서, 수치적 현재 콘텍스트 값이 "유효" 콘텍스트 값(또는 "유효" 상태 값)이 아니라고 확인되면, 해시 테이블("ari_hash_m")의 평가로부터 획득된 해시 테이블 인덱스 값(예를 들어, i1-1, i1, 또는 i2)이 수치적 콘텍스트 값들의 구간 932, 934, 936과 연관된 맵핑 규칙 인덱스 값을 획득하기 위해 이용될 수 있다. 예를 들어, 해시 테이블 인덱스 값(예를 들어, i1-1, i1, 또는 i2)은 수치적 현재 콘텍스트 값이 있는 구간 932, 934, 936과 연관된 맵핑 규칙 인덱스 값을 기술하는 추가 맵핑 테이블(예를 들어, "ari_lookup_m")의 엔트리를 지칭하는데 이용될 수 있다.

[0054] 더 자세한 사항들을 위해, 알고리즘 "arith_get_pk"에 대한 상세한 논의가 하기에서 언급된다(여기서 이 알고리즘 "arith_get_pk"에 대한 각각 다른 선택사항들이 있는데, 그 예시들이 도 5e 및 5f에 도시된다).

[0055] 또한, 구간들의 크기는 각각의 경우마다 각각 다를 수 있음을 알아야 한다. 몇몇 경우들에서, 수치적 콘텍스트 값들의 구간은 단일 수치적 콘텍스트 값을 포함한다. 그러나, 많은 경우들에서, 구간은 복수의 수치적 콘텍스트 값들을 포함할 수 있다.

[0056] 4. 도 10에 따른 오디오 인코더

[0057] 도 10은 본 발명의 일 실시예에 따른 오디오 인코더(1000)에 대한 블록 도식도를 도시한다. 도 10에 따른 오디오 인코더(1000)는 도 7에 따른 오디오 인코더(700)와 유사하여, 동일한 신호들 및 수단들은 도 7 및 10에서 동일한 도면 부호들로 지칭된다.

[0058] 오디오 인코더(1000)는 입력된 오디오 정보(710)를 수신하고, 그에 기초하여, 인코딩된 오디오 정보(712)를 제공하도록 구성된다. 오디오 인코더(1000)는 입력된 오디오 정보(710)의 시간 도메인 표현에 기초하여 주파수 도메인 표현(722)을 제공하도록 구성되는 에너지 압축 시간 도메인 대 주파수 도메인 변환기(720)를 포함하여, 주파수 도메인 오디오 표현(722)이 스펙트럼 값들의 셋트를 포함한다. 오디오 인코더(1000)는 또한, (예를 들어, 복수의 가변 길이 코드워드들을 포함할 수 있는) 인코딩된 오디오 정보(712)를 획득하기 위해 가변 길이 코드워드를 이용하여, (주파수 도메인 오디오 표현(722)을 형성하는 스펙트럼 값들의 셋트 중에서) 하나의 스펙트럼 값, 또는 그의 전처리된 버전을 인코딩하도록 구성되는 산술 인코더(1030)를 포함한다.

[0059] 산술 인코더(1030)는 콘텍스트 상태에 따라 코드 값(즉, 가변 결이 코드워드)에, 스펙트럼 값, 또는 복수의 스펙트럼 값들, 또는 스펙트럼 값이나 복수의 스펙트럼 값들의 최상위 비트 평면의 값을 맵핑하도록 구성된다. 산술 인코더(1030)는 콘텍스트 상태에 따라 코드 값으로의, 스펙트럼 값, 또는 복수의 스펙트럼 값들, 또는 스펙트럼 값이나 복수의 스펙트럼 값들의 최상위 비트 평면의 맵핑을 기술하는 맵핑 규칙을 선택하도록 구성된다. 산술 인코더는 복수의 이전에 인코딩된 (바람직하게는, 그러나 반드시 인접하지는 않는) 스펙트럼 값들에 따라 현재 콘텍스트 상태를 결정하도록 구성된다. 이를 위해, 산술 인코더는, (예를 들어, 상응하는 맵핑 규칙을 선택하기 위하여) 인코딩되는 하나 이상의 스펙트럼 값들과 연관된 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 콘텍스트 서브구역 값에 따라, (예를 들어, 상응하는 맵핑 규칙을 선택하기 위하여) 인코딩되는 하나 이상의 이전에 인코딩된 스펙트럼 값들과 연관된 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값의 수치 표현을 수정하도록 구성된다.

[0060] 알 수 있는 바와 같이, 코드 값으로의, 스펙트럼 값, 또는 복수의 스펙트럼 값들, 또는 스펙트럼 값이나 복수의 스펙트럼 값들의 최상위 비트 평면의 맵핑은 맵핑 규칙 정보(742)에 의해 기술된 맵핑 규칙을 이용하여 스펙트럼 값 인코딩(740)에 의해 수행될 수 있다. 상태 추적기(750)는 콘텍스트 상태를 추적하도록 구성될 수 있다.

상태 추적기(750)는, 인코딩되는 하나 이상의 스펙트럼 값들의 인코딩과 연관된 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 콘텍스트 서브구역 값에 따라, 하나 이상의 이전에 인코딩된 스펙트럼 값들의 인코딩과 연관된 콘텍스트 상태를 기술하는 수치적 이전 콘텍스트 값의 수치 표현을 수정하도록 구성될 수 있다. 수치적 이전 콘텍스트 값의 수치 표현의 수정은, 예를 들어, 수치적 이전 콘텍스트 값 및 하나 이상의 콘텍스트 서브구역 값들을 수신하고 수치적 현재 콘텍스트 값을 제공하는 수치 표현 수정기(1052)에 의해 수행될 수 있다. 이에 따라, 상태 추적기(1050)는, 예를 들어, 수치적 현재 콘텍스트 값의 형태로, 현재 콘텍스트 상태를 기술하는 정보(754)를 제공한다. 맵핑 규칙 선택기(1060)는, 코드 값으로의, 스펙트럼 값, 또는 복수의 스펙트럼 값들, 또는 스펙트럼 값이나 복수의 스펙트럼 값들의 최상위 비트 평면의 맵핑을 기술하는 맵핑 규칙, 예를 들어, 누적 빈도 테이블을 선택할 수 있다. 이에 따라, 맵핑 규칙 선택기(1060)는 스펙트럼 인코딩(740)에 맵핑 규칙 정보(742)를 제공한다.

[0061] 몇몇 실시예들에서, 상태 추적기 1050은 상태 추적기 750 또는 상태 추적기 826과 동일할 수 있음을 알아야 한다. 맵핑 규칙 선택기 1060은, 몇몇 실시예들에서, 맵핑 규칙 선택기 760, 또는 맵핑 규칙 선택기 828과 동일할 수 있음을 또한 알아야 한다.

[0062] 상기를 요약하면, 오디오 인코더(1000)는 시간 도메인 대 주파수 도메인 변환기에 의해 제공된 주파수 도메인 오디오 표현의 산술 인코딩을 수행한다. 산술 인코딩은 콘텍스트에 따르므로, 맵핑 규칙(예를 들어, 누적 빈도 테이블)이 이전에 인코딩된 스펙트럼 값들에 따라 선택된다. 이에 따라, 시간 및/또는 주파수에서 (또는 적어도 미리 결정된 환경 내에서) 서로 및/또는 현재 인코딩된 스펙트럼 값(즉, 현재 인코딩된 스펙트럼 값의 미리 결정된 환경 내의 스펙트럼 값들)에 인접한 스펙트럼 값들이 산술 인코딩에 의해 평가된 확률 분포를 조절하도록 산술 인코딩에서 고려된다.

[0063] 수치적 현재 콘텍스트 값을 결정할 때, 하나 이상의 이전에 인코딩된 스펙트럼 값들과 연관된 콘텍스트 상태를 기술하는 수치적 이전 콘텍스트 값의 수치 표현은, 인코딩되는 하나 이상의 스펙트럼 값들과 연관된 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 콘텍스트 서브구역 값에 따라 수정된다. 이러한 접근법은 수치적 현재 콘텍스트 값의 전적인 재계산을 피하는 것을 가능하게 하는데, 이 전적인 재계산은 종래의 접근법들에서 자원의 상당한 양을 소비한다. 수치적 이전 콘텍스트 값의 수치 표현의 재스케일링의 결합, 콘텍스트 서브구역 값 또는 그로부터 도출되는 값을 수치적 이전 콘텍스트 값의 수치 표현 또는 수치적 이전 콘텍스트 값의 처리된 수치 표현에 추가, 콘텍스트 서브구역 값에 따라 수치적 이전 콘텍스트 값의 (전체 수치 표현보다는) 수치 표현의 일부분의 대체, 등등을 포함하는 수치적 이전 콘텍스트 값의 수치 표현의 수정에 대한 다양한 가능성이 존재한다. 그러므로, 일반적으로 수치적 현재 콘텍스트 값의 수치 표현은 수치적 이전 콘텍스트 값의 수치 표현에 기초하고 또한 적어도 하나의 콘텍스트 서브구역 값에 기초하여 획득되는데, 여기서, 일반적으로, 예를 들어, 덧셈 연산, 뺄셈 연산, 곱셈 연산, 나눗셈 연산, 부울(Boolean) AND 연산, 부울 OR 연산, 부울 NAND 연산, 부울 NOR 연산, 부울 부정 연산, 보수(complement) 연산, 또는 이동(shift) 연산 중에 두 개 이상의 연산과 같은 연산들의 결합은 수치적 이전 콘텍스트 값을 콘텍스트 서브구역 값과 결합하도록 수행된다. 이에 따라, 수치적 이전 콘텍스트 값으로부터 수치적 현재 콘텍스트 값을 도출할 때, 수치적 이전 콘텍스트 값의 수치 표현의 적어도 일부분이 일반적으로 (각각 다른 위치로의 선택적 이동를 제외하고) 달라지지 않은 채로 유지된다. 그에 반해서, 수치적 이전 콘텍스트 값의 수치 표현의 다른 부분들은 하나 이상의 콘텍스트 서브구역 값들에 따라 달라진다. 그러므로, 수치적 현재 콘텍스트 값의 전적인 재계산을 피하면서, 수치적 현재 콘텍스트 값이 비교적 적은 계산 노력으로 획득될 수 있다.

[0064] 이렇게 하여, 맵핑 규칙 선택기(1060)에 의해 사용되기에 아주 적합한 의미 있는 수치적 현재 콘텍스트 값이 획득될 수 있다.

[0065] 결과적으로, 콘텍스트 계산을 충분히 간단하게 유지함으로써 효율적인 인코딩이 달성될 수 있다.

[0066] 5. 도 11에 따른 오디오 디코더

[0067] 도 11은 오디오 디코더(1100)에 대한 블록 도식도를 도시한다. 오디오 디코더 1100은 도 8에 따른 오디오 디코더 800과 유사하여, 동일한 신호들, 수단들, 및 기능들은 동일한 도면 부호들로 지칭된다.

[0068] 오디오 디코더(1100)는 인코딩된 오디오 정보(810)를 수신하고, 그에 기초하여, 디코딩된 오디오 정보(812)를 제공하도록 구성된다. 오디오 디코더(1100)는 스펙트럼 값들의 산술적으로 인코딩된 표현(821)에 기초하여 복수의 디코딩된 스펙트럼 값들(822)을 제공하도록 구성되는 산술 디코더(1120)를 포함한다. 오디오 디코더(1100)는 또한, 디코딩된 스펙트럼 값들(822)을 수신하고, 디코딩된 오디오 정보(812)를 획득하기 위해, 디코딩된 스펙트럼 값들(822)을 이용하여, 디코딩된 오디오 정보를 이룰 수 있는 시간 도메인 오디오 표현(812)을 제공하도록 구성되는 주파수 도메인 대 시간 도메인 변환기(830)를 포함한다.

[0069] 산술 디코더(1120)는 하나 이상의 디코딩된 스펙트럼 값들, 또는 하나 이상의 디코딩된 스펙트럼 값들의 적어도 일부분(예를 들어, 최상위 비트 평면)을 표현하는 심볼 코드에 스펙트럼 값들의 산술적으로 인코딩된 표현(821)의 코드 값을 맵핑하도록 구성되는 스펙트럼 값 결정기(824)를 포함한다. 스펙트럼 값 결정기(824)는 맵핑 규칙 정보(828a)에 의해 기술될 수 있는 맵핑 규칙에 따라 맵핑을 수행하도록 구성될 수 있다. 맵핑 규칙 정보(828a)는, 예를 들어, 맵핑 규칙 인덱스 값을 포함할 수 있거나, 누적 빈도 테이블의 엔트리들 중 선택된 셋트를 포함할 수 있다.

[0070] 산술 디코더(1120)는, 콘텍스트 상태가 콘텍스트 상태 정보(1126a)에 의해 기술될 수 있는 콘텍스트 상태에 따라, (하나 이상의 스펙트럼 값들을 기술하는) 심볼 코드로의, (스펙트럼 값들의 산술적으로 인코딩된 표현(821)에 의해 기술된) 코드 값의 맵핑을 기술하는 맵핑 규칙(예를 들어, 누적 빈도 테이블)을 선택하도록 구성된다. 콘텍스트 상태 정보(1126a)는 수치적 현재 콘텍스트 값의 형태를 취할 수 있다. 산술 디코더(1120)는 복수의 이전에 디코딩된 스펙트럼 값들(822)에 따라 현재 콘텍스트 상태를 결정하도록 구성된다. 이를 위해, 이전에 디코딩된 스펙트럼 값들을 기술하는 정보를 수신하는 상태 추적기(1126)가 이용될 수 있다. 산술 디코더는, 디코딩되는 하나 이상의 스펙트럼 값들과 연관된 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 콘텍스트 서브구역 값에 따라, 하나 이상의 이전에 디코딩된 스펙트럼 값들과 연관된 콘텍스트 상태를 기술하는 수치적 이전 콘텍스트 값의 수치 표현을 수정하도록 구성된다. 수치적 이전 콘텍스트 값의 수치 표현의 수정은, 예를 들어, 상태 추적기(1126)의 일부분인 수치 표현 수정기(1127)에 의해 수행될 수 있다. 이에 따라, 예를 들어, 수치적 현재 콘텍스트 값의 형태로, 현재 콘텍스트 상태 정보(1126a)가 획득된다. 맵핑 규칙의 선택은, 현재 콘텍스트 상태 정보(1126a)로부터 맵핑 규칙 정보(828a)를 도출하여, 스펙트럼 값 결정기(824)에 맵핑 규칙 정보(828a)를 제공하는 맵핑 규칙 선택기(1128)에 의해 수행될 수 있다.

[0071] 오디오 신호 디코더(1100)의 기능과 관련하여, 맵핑 규칙이 현재 콘텍스트 상태에 따라 선택되기 때문에, 이는, 결국, 복수의 이전에 디코딩된 스펙트럼 값들에 따라 결정되므로, 산술 디코더(1120)는 디코딩되는 스펙트럼 값에, 대체로, 잘 적응되는 맵핑 규칙(예를 들어, 누적 빈도 테이블)을 선택하도록 구성된다는 것을 알아야 한다. 이에 따라, 디코딩되는 인접한 스펙트럼 값들 사이의 통계적 의존성이 활용될 수 있다.

[0072] 또한, 디코딩되는 하나 이상의 스펙트럼 값들의 디코딩과 연관된 콘텍스트 상태를 기술하는 수치적 현재 콘텍스트 값의 수치 표현을 획득하기 위해, 콘텍스트 서브구역 값에 따라, 하나 이상의 이전에 디코딩된 스펙트럼 값들의 디코딩과 연관된 콘텍스트 상태를 기술하는 수치적 이전 콘텍스트 값의 수치 표현을 수정하여, 비교적 적은 계산 노력으로, 맵핑 규칙 인덱스 값에 맵핑하는데 아주 적합한 현재 콘텍스트 상태에 관한 의미 있는 정보를 획득하는 것이 가능하다. (아마도 비트 이동되거나 스케일링된 버전에서) 수치적 이전 콘텍스트 값의 수치 표현의 적어도 일부분을 유지하고, 한편 수치적 이전 콘텍스트 값에서는 고려되지 않았지만 수치적 현재 콘텍스트 값에서는 고려되어야 하는 콘텍스트 서브구역 값들에 따라 수치적 이전 콘텍스트 값의 수치 표현의 다른 부분을 업데이트하여, 수치적 현재 콘텍스트 값을 도출하기 위한 연산들의 수가 상당히 작게 유지될 수 있다. 또

한, 인접한 스펙트럼 값들을 디코딩하기 위해 이용된 콘텍스트들이 일반적으로 유사하거나 서로 연관되어 있다는 사실을 활용할 수 있다. 예를 들어, 제1 스펙트럼 값(또는 제1 복수의 스펙트럼 값들)의 디코딩을 위한 콘텍스트는 이전에 디코딩된 스펙트럼 값들의 제1 셋트에 따른다. 제1 스펙트럼 값(또는 스펙트럼 값들의 제1 셋트)에 인접하는 제2 스펙트럼 값(또는 스펙트럼 값들의 제2 셋트)의 디코딩을 위한 콘텍스트는 이전에 디코딩된 스펙트럼 값들의 제2 셋트를 포함할 수 있다. 제1 스펙트럼 값 및 제2 스펙트럼 값이 (예를 들어, 연관된 주파수들에 대하여) 인접하는 것으로 추정됨에 따라, 제1 스펙트럼 값의 코딩을 위한 콘텍스트를 결정하는 스펙트럼 값들의 제1 셋트는 제2 스펙트럼 값의 디코딩을 위한 콘텍스트를 결정하는 스펙트럼 값들의 제2 셋트와의 약간의 중첩을 포함할 수 있다. 이에 따라, 제2 스펙트럼 값의 디코딩을 위한 콘텍스트 상태가 제1 스펙트럼 값의 디코딩을 위한 콘텍스트 상태와의 약간의 연관성을 포함한다는 것이 쉽게 이해될 수 있다. 콘텍스트 도출, 즉, 수치적 현재 콘텍스트 값 도출의 계산 효율성이 그러한 연관성을 이용하여 달성될 수 있다. (예를 들어, 수치적 이전 콘텍스트 값에 의해 기술된 콘텍스트 상태와 수치적 현재 콘텍스트 값에 의해 기술된 콘텍스트 상태 사이) 인접한 스펙트럼 값들의 디코딩을 위한 콘텍스트 상태들 사이의 연관성은, 수치적 이전 콘텍스트 상태의 도출을 위해 고려되지 않는 콘텍스트 서브구역 값들을 따르는 수치적 이전 콘텍스트 값의 그러한 부분들만 수정하고, 수치적 이전 콘텍스트 값으로부터 수치적 현재 콘텍스트 값을 도출하여 효율적으로 활용될 수 있는 것으로 확인됐다.

[0073] 결론은, 여기에 기술된 구상들은 수치적 현재 콘텍스트 값을 도출할 때 특히 좋은 계산 효율성을 가능하게 한다.

[0074] 더 자세한 사항들이 하기에서 기술될 것이다.

[0075] 6. 도 12에 따른 오디오 인코더

[0076] 도 12는 본 발명의 일 실시예에 따른 오디오 인코더에 대한 블록 도식도를 도시한다. 도 12에 따른 오디오 인코더 1200은 도 7에 따른 오디오 인코더 700과 유사하여, 동일한 수단들, 신호들 및 기능들은 동일한 도면 부호들로 지칭된다.

[0077] 오디오 인코더(1200)는 입력된 오디오 정보(710)를 수신하고, 그에 기초하여, 인코딩된 오디오 정보(712)를 제공하도록 구성된다. 오디오 인코더(1200)는 입력된 오디오 정보(710)의 시간 도메인 오디오 표현에 기초하여 주파수 도메인 오디오 표현(722)을 제공하도록 구성되는 에너지 압축 시간 도메인 대 주파수 도메인 변환기(720)를 포함하여, 주파수 도메인 오디오 표현(722)이 스펙트럼 값들의 셋트를 포함한다. 오디오 인코더(1200)는 또한, (예를 들어, 복수의 가변 길이 코드워드들을 포함할 수 있는) 인코딩된 오디오 정보(712)를 획득하기 위해 가변 길이 코드워드를 이용하여, (주파수 도메인 오디오 표현(722)을 형성하는 스펙트럼 값들의 셋트 중에서) 하나의 스펙트럼 값, 또는 복수의 스펙트럼 값들, 또는 그의 전처리된 버전을 인코딩하도록 구성되는 산술 인코더(1230)를 포함한다.

[0078] 산술 인코더(1230)는, 콘텍스트 상태에 따라, 코드 값(즉, 가변 길이 코드워드)에, 스펙트럼 값, 또는 복수의 스펙트럼 값들, 또는 스펙트럼 값이나 복수의 스펙트럼 값들의 최상위 비트 평면의 값을 맵핑하도록 구성된다. 산술 인코더(1230)는, 콘텍스트 상태에 따라, 코드 값으로의, 스펙트럼 값, 또는 복수의 스펙트럼 값들, 또는 스펙트럼 값이나 복수의 스펙트럼 값의 최상위 비트 평면의 맵핑을 기술하는 맵핑 규칙을 선택하도록 구성된다. 산술 인코더는 복수의 이전에 인코딩된 (바람직하게는, 그러나 반드시 그렇지는 않는, 인접한) 스펙트럼 값들에 따라 현재 콘텍스트 상태를 결정하도록 구성된다. 이를 위해, 산술 인코더는 이전에 인코딩된 스펙트럼 값들에 기초하여 복수의 콘텍스트 서브구역 값들을 획득하며, 상기 콘텍스트 서브구역 값을 저장하고, 저장된 콘텍스트 서브구역 값들에 따라 인코딩되는 하나 이상의 스펙트럼 값들과 연관된 수치적 현재 콘텍스트 값을 도출하도록 구성된다. 또한, 산술 인코더는, 복수의 이전에 인코딩된 스펙트럼 값들과 연관된 공통 콘텍스트 서브구역 값을

획득하기 위해, 복수의 이전에 인코딩된 스펙트럼 값들에 의해 형성된 벡터의 놈(norm)을 계산하도록 구성된다.

[0079] 알 수 있는 바와 같이, 코드 값으로의, 스펙트럼 값, 또는 복수의 스펙트럼 값들, 또는 스펙트럼 값이나 복수의 스펙트럼 값들의 최상위 비트 평면의 맵핑은 맵핑 규칙 정보(742)에 의해 기술된 맵핑 규칙을 이용하여 스펙트럼 값 인코딩(740)에 의해 수행될 수 있다. 상태 추적기(1250)는 콘텍스트 상태를 추적하도록 구성될 수 있고, 복수의 이전에 인코딩된 스펙트럼 값들과 연관된 공통 콘텍스트 서브구역 값들을 획득하기 위해, 복수의 이전에 인코딩된 스펙트럼 값들에 의해 형성된 벡터의 놈을 계산하기 위한 콘텍스트 서브구역 값 계산기(computer, 1252)를 포함할 수 있다. 상태 추적기(1250)는 또한 바람직하게는 콘텍스트 서브구역 값 계산기(1252)에 의해 수행된 콘텍스트 서브구역 값의 상기 계산 결과에 따라 현재 콘텍스트 상태를 결정하도록 구성된다. 이에 따라, 상태 추적기(1250)는 현재 콘텍스트 상태를 기술하는 정보(1254)를 제공한다. 맵핑 규칙 선택기(1260)는, 코드 값으로의, 스펙트럼 값, 또는 스펙트럼 값의 최상위 비트 평면의 맵핑을 기술하는 맵핑 규칙, 예를 들어, 누적 빈도 테이블을 선택할 수 있다. 이에 따라, 맵핑 규칙 선택기(1260)는 스펙트럼 인코딩(740)에 맵핑 규칙 정보(742)를 제공한다.

[0080] 상기를 요약하면, 오디오 인코더(1200)는 시간 도메인 대 주파수 도메인 변환기(720)에 의해 제공된 주파수 도메인 오디오 표현의 산술 인코딩을 수행한다. 산술 인코딩이 콘텍스트에 따르므로, 맵핑 규칙(즉, 누적 빈도 테이블)이 이전에 인코딩된 스펙트럼 값들에 따라 선택된다. 이에 따라, 서로 및/또는 현재 인코딩된 스펙트럼 값(즉, 현재 인코딩된 스펙트럼 값의 미리 결정된 환경 내의 스펙트럼 값들)에 시간 및/또는 주파수(또는, 적어도 미리 결정된 환경 내)에서 인접한 스펙트럼 값들은 산술 인코딩에 의해 평가된 확률 분포를 조절하기 위해 산술 인코딩에서 고려된다.

[0081] 수치적 현재 콘텍스트 값을 제공하기 위해, 복수의 이전에 인코딩된 스펙트럼 값들과 연관된 콘텍스트 서브구역 값이 복수의 이전에 인코딩된 스펙트럼 값들에 의해 형성된 벡터의 놈 계산에 기초하여 획득된다. 수치적 현재 콘텍스트 값의 결정에 대한 결과는 현재 콘텍스트 상태의 선택, 즉, 맵핑 규칙의 선택에 적용된다.

[0082] 복수의 이전에 인코딩된 스펙트럼 값들에 의해 형성된 벡터의 놈을 계산하여, 인코딩되는 하나 이상의 스펙트럼 값들의 콘텍스트의 일부분을 기술하는 의미 있는 정보가 얻어지는데, 여기서 이전에 인코딩된 스펙트럼 값들의 벡터의 놈은 일반적으로 비교적 적은 비트 수로 표현될 수 있다. 그러므로, 수치적 현재 콘텍스트 값의 도출에서 추후 사용되기 위해 저장되어야 하는 콘텍스트 정보의 양이, 콘텍스트 서브구역 값의 계산을 위해 상기에서 논의된 접근법을 적용하여 상당히 작게 유지될 수 있다. 이전에 인코딩된 스펙트럼 값들의 벡터의 놈은 일반적으로 콘텍스트의 상태에 관한 가장 유효한 정보를 포함하는 것으로 확인됐다. 그에 반해, 상기 이전에 인코딩된 스펙트럼 값들의 부호는 콘텍스트의 상태에 대한 부차적인 영향을 포함하여, 추후 사용을 위해 저장되는 정보의 양을 줄이기 위해 이전에 디코딩된 스펙트럼 값들의 부호를 무시하는 것이 맞다는 것으로 확인됐다. 또한, 일반적으로 놈의 계산에 의해 얻어지는 평균화하는 효과는 실질적으로 영향을 받지 않는 콘텍스트 상태에 관한 가장 중요한 정보를 남기므로, 이전에 인코딩된 스펙트럼 값들의 벡터의 놈 계산이 콘텍스트 서브구역 값의 도출을 위한 타당한 접근법인 것으로 확인됐다. 요약하면, 콘텍스트 서브구역 값 계산기(1252)에 의해 수행된 콘텍스트 서브구역 값 계산은 저장 및 추후 재사용을 위해 압축된(compact) 콘텍스트 서브구역 정보를 제공하는 것을 가능하게 하는데, 여기서 콘텍스트 상태에 관한 가장 관련있는 정보는 정보량의 감소에도 불구하고 보존된다.

[0083] 이에 따라, 입력된 오디오 정보(710)에 대한 효율적인 인코딩이 달성될 수 있는 한편, 계산 노력 및 산술 인코더(1230)에 의해 저장되는 데이터의 양이 상당히 적게 유지된다.

[0084] 7. 도 13에 따른 오디오 디코더

[0085] 도 13은 오디오 디코더(1300)에 대한 블록 도식도를 도시한다. 오디오 디코더 1300이 도 8에 따른 오디오 디코

더 800, 및 도 11에 따른 오디오 디코더 1100와 유사하기 때문에, 동일한 수단, 신호들, 및 방법들은 동일한 도면 부호들로 지칭된다.

[0086] 오디오 디코더(1300)는 인코딩된 오디오 정보(810)를 수신하고, 그에 기초하여, 디코딩된 오디오 정보(812)를 제공하도록 구성된다. 오디오 디코더(1300)는 스펙트럼 값들의 산술적으로 인코딩된 표현(821)에 기초하여 복수의 디코딩된 스펙트럼 값들(822)를 제공하도록 구성되는 산술 디코더(1320)를 포함한다. 오디오 디코더(1300)는 또한, 디코딩된 스펙트럼 값들(822)을 수신하고, 디코딩된 오디오 정보(812)를 획득하기 위해, 디코딩된 스펙트럼 값들(822)을 이용하여, 디코딩된 오디오 정보를 이룰 수 있는 시간 도메인 오디오 표현(812)을 제공하도록 구성되는 주파수 도메인 대 시간 도메인 변환기(830)를 포함한다.

[0087] 산술 디코더(1320)는, 하나 이상의 디코딩된 스펙트럼 값들, 또는 하나 이상의 디코딩된 스펙트럼 값들의 적어도 일부분(예를 들어, 최상위 비트 평면)을 표현하는 심볼 코드에, 스펙트럼 값들의 산술적으로 인코딩된 표현(821)의 코드 값을 맵핑하도록 구성되는 스펙트럼 값 결정기(824)를 포함한다. 스펙트럼 값 결정기(824)는 맵핑 규칙 정보(828a)에 의해 기술되는 맵핑 규칙에 따라 맵핑을 수행하도록 구성될 수 있다. 맵핑 규칙 정보(828a)는, 예를 들어, 맵핑 규칙 인덱스 값, 또는 누적 빈도 테이블의 엔트리들 중 선택된 셋트를 포함할 수 있다.

[0088] 산술 디코더(1320)는 (콘텍스트 상태 정보(1326a)에 의해 기술될 수 있는) 콘텍스트 상태에 따라 (하나 이상의 스펙트럼 값들을 기술하는) 심볼 코드로의 (스펙트럼 값들의 산술적으로 인코딩된 표현(821)에 의해 기술된) 코드 값의 맵핑을 기술하는 맵핑 규칙(예를 들어, 누적 빈도 테이블)을 선택하도록 구성된다. 산술 디코더(1320)는 복수의 이전에 디코딩된 스펙트럼 값들(822)에 따라 현재 콘텍스트 상태를 결정하도록 구성된다. 이를 위해, 이전에 디코딩된 스펙트럼 값들을 기술하는 정보를 수신하는 상태 추적기(1326)가 이용될 수 있다. 산술 디코더는 또한 이전에 디코딩된 스펙트럼 값들에 기초해 복수의 콘텍스트 서브구역 값들을 획득하여, 상기 콘텍스트 서브구역 값들을 저장하도록 구성된다. 산술 디코더는 저장된 콘텍스트 서브구역 값들에 따라 디코딩되는 하나 이상의 스펙트럼 값들과 연관된 수치적 현재 콘텍스트 값을 도출하도록 구성된다. 산술 디코더(1320)는, 복수의 이전에 디코딩된 스펙트럼 값들과 연관된 공통 콘텍스트 서브구역 값을 획득하기 위해, 복수의 이전에 디코딩된 스펙트럼 값들에 의해 형성된 벡터의 놈을 계산하도록 구성된다.

[0089] 복수의 이전에 인코딩된 스펙트럼 값들에 의해 형성된 벡터의 놈 계산은, 복수의 이전에 디코딩된 스펙트럼 값들과 연관된 공통 콘텍스트 서브구역 값을 획득하기 위해, 예를 들어, 콘텍스트 추적기(1326)의 일부분인 콘텍스트 서브구역 값 계산기(1327)에 의해 수행될 수 있다. 이에 따라, 현재 콘텍스트 상태 정보(1326a)가 콘텍스트 서브구역 값들에 기초하여 획득되는데, 여기서 상태 추적기(1326)는 바람직하게는 저장된 콘텍스트 서브구역 값들에 따라 디코딩되는 하나 이상의 스펙트럼 값들과 연관된 수치적 현재 콘텍스트 값을 제공한다. 맵핑 규칙들의 선택은 현재 콘텍스트 상태 정보(1326a)로부터 맵핑 규칙 정보(828a)를 도출하여, 스펙트럼 값 결정기(824)에 맵핑 규칙 정보(828a)를 제공하는 맵핑 정보 선택기(1328)에 의해 수행될 수 있다.

[0090] 오디오 신호 디코더(1300)의 기능과 관련하여, 맵핑 규칙이 현재 콘텍스트 상태에 따라 선택되어, 결국, 복수의 이전에 디코딩된 스펙트럼 값들에 따라 결정되기 때문에, 산술 디코더(1320)는 디코딩되는 스펙트럼 값에 대체로 잘 적응되는 맵핑 규칙(예를 들어, 누적 빈도 테이블)을 선택하도록 구성된다는 것을 알아야 한다. 이에 따라, 디코딩되는 인접한 스펙트럼 값들 사이의 통계적 의존성이 활용될 수 있다.

[0091] 그러나, 수치적 현재 값의 결정에서 추후 사용하기 위해, 복수의 이전에 디코딩된 스펙트럼 값들로 형성된 벡터의 놈 계산에 기초하는 콘텍스트 서브구역 값들을 저장하는 것이, 메모리 사용의 측면에서, 효율적인 것으로 확인됐다. 또한 그러한 콘텍스트 서브구역 값들이 여전히 가장 관련 있는 콘텍스트 정보를 포함하고 있는 것으로 확인됐다. 이에 따라, 상태 추적기(1326)에 의해 이용된 구상은 코딩 효율성, 계산 효율성, 및 저장 효율성 사이에 좋은 절충을 이룬다.

[0092] 더 자세한 사항들이 하기에서 기술될 것이다.

[0093] 8. 도 1에 따른 오디오 인코더

[0094] 다음에서는, 본 발명의 일 실시예에 따른 오디오 인코더가 기술될 것이다. 도 1은 그러한 오디오 인코더(100)에 대한 블록 도식도를 도시한다.

[0095] 오디오 인코더(100)는 입력된 오디오 정보(100)를 수신하고, 그에 기초하여, 인코딩된 오디오 정보를 이루는 비트스트림(bitstream, 112)를 제공하도록 구성된다. 오디오 인코더(100)는, 입력된 오디오 신호(110)를 수신하고, 그에 기초하여, 전처리된 입력된 오디오 정보(110a)를 제공하도록 구성되는 전처리기(120)를 선택적으로 포함한다. 오디오 인코더(100)는 또한, 신호 변환기로라고도 지칭되는 에너지 압축 시간 도메인 대 주파수 도메인 신호 전환기(130)를 포함한다. 신호 변환기(130)는, 입력된 오디오 정보(110, 110a)를 수신하고, 그에 기초하여, 바람직하게는 스펙트럼 값들의 셋트 형태를 취하는 주파수 도메인 오디오 정보(132)를 제공하도록 구성된다. 예를 들어, 신호 전환기(130)는 입력된 오디오 신호(110, 110a)의 프레임(예를 들어, 시간 도메인 샘플들의 블록)을 수신하여, 각각의 오디오 프레임의 오디오 콘텐츠를 표현하는 스펙트럼 값들의 셋트를 제공하도록 구성될 수 있다. 또한, 신호 전환기(130)는, 복수의 서브시퀀스(subsequence), 중첩 또는 비 중첩, 입력된 오디오 정보(110, 110a)의 오디오 프레임들을 수신하고, 그에 기초하여, 스펙트럼 값들의 서브시퀀스 셋트들, 각각의 프레임과 연관된 스펙트럼 값들의 하나의 셋트의 시퀀스를 포함하는 시간 주파수 도메인 오디오 표현을 제공하도록 구성될 수 있다.

[0096] 에너지 압축 시간 도메인 대 주파수 도메인 신호 전환기(130)는 각각 다른, 중첩되거나 중첩되지 않은, 주파수 범위들과 연관된 스펙트럼 값들을 제공하는 에너지 압축 필터뱅크(filterbank)를 포함할 수 있다. 예를 들어, 신호 전환기(130)는 전환 윈도우(window)를 이용하여 입력된 오디오 정보(110, 110a)(또는, 그의 프레임)를 윈도윙하고, 윈도윙된 입력된 오디오 정보(110, 110a)(또는, 그의 윈도윙된 프레임)의 변형 이산 코사인 변환을 수행하도록 구성되는 윈도윙 MDCT 전환기(130a)를 포함할 수 있다. 이에 따라, 주파수 도메인 오디오 표현(132)은, 예를 들어, 입력된 오디오 정보의 프레임과 연관된 MDCT 계수들의 형태로 1024개의 스펙트럼 값들의 셋트를 포함할 수 있다.

[0097] 오디오 인코더(100)는 주파수 도메인 오디오 표현(132)을 수신하고, 그에 기초하여, 후처리된 주파수 도메인 오디오 표현(142)을 제공하도록 구성되는 스펙트럼 후처리기(140)를 추가하여 선택적으로 포함할 수 있다. 스펙트럼 후처리(140)는, 예를 들어, 일시적(temporal) 잡음 정형(shaping) 및/또는 장기 예측 및/또는 종래 기술에서 알려지 임의의 다른 스펙트럼 후처리를 수행하도록 구성될 수 있다. 오디오 인코더는 주파수 도메인 오디오 표현(132) 또는 그것의 후처리된 버전(142)을 수신하고, 스케일링되고 양자화된 주파수 도메인 오디오 표현(152)을 제공하도록 구성되는 스케일러/양자화기(150)를 추가하여 선택적으로 포함한다.

[0098] 오디오 인코더는(100), 입력된 오디오 정보(100)(또는 그것의 후처리된 버전(110a))을 수신하고, 그에 기초하여, 에너지 압축 시간 도메인 대 주파수 도메인 신호 전환기(130)의 제어, 선택적 스펙트럼 후처리기(140)의 제어, 및/또는 선택적 스케일러/양자화기(150)의 제어에 이용될 수 있는 선택적 제어 정보를 제공하도록 구성되는 심리 음향적 모델 처리기(160)를, 선택적으로, 추가하여 포함한다. 예를 들어, 심리 음향적 모델 처리기(160)는 입력된 오디오 정보(110, 110a) 중 어떤 구성요소들이 오디오 콘텐츠에 대한 인간의 지각에 특히 중요하고 입력된 오디오 정보(110, 110a) 중 어떤 구성요소들이 오디오 콘텐츠의 지각에 덜 중요한지를 결정하기 위해 입력된 오디오 정보를 분석하도록 구성될 수 있다. 이에 따라, 심리 음향적 모델 처리기(160)는 스케일러/양자화기(150)에 의한 주파수 도메인 오디오 표현(132, 142)의 스케일링 및/또는 스케일러/양자화기(150)에 의해 적용된 양자화 분해능(resolution)을 조절하기 위해 오디오 인코더(100)에 의해 이용되는 제어 정보를 제

공할 수 있다. 결과적으로, 지각적으로 중요한 스케일링 인자 대역들(즉, 오디오 콘텐츠에 대한 인간의 지각에 특히 중요한 인접한 스펙트럼 값들의 그룹들)이 큰 스케일링 인자로 스케일링되고 비교적 높은 분해능으로 양자화되는데 반해, 지각적으로 덜 중요한 스케일링 인자 대역들(즉, 인접한 스펙트럼 값들의 그룹들)은 비교적 작은 스케일링 인자로 스케일링되고 비교적 낮은 양자화 분해능으로 양자화된다. 이에 따라, 지각적으로 더 중요한 주파수들의 스케일링된 스펙트럼 값들이 지각적으로 덜 중요한 주파수들의 스펙트럼 값들보다 일반적으로 상당히 더 크다.

[0099] 오디오 인코더는 또한, 주파수 도메인 오디오 표현(132)의 스케일링되고 양자화된 버전(152)(또는, 그렇지 않으면, 주파수 도메인 오디오 표현(132)의 후처리된 버전(142), 또는 심지어 주파수 도메인 오디오 표현(132) 그 자체)을 수신하고, 그에 기초하여, 산술 코드워드 정보(172a)를 제공하도록 구성되는 산술 인코더(170)를 포함하여, 산술 코드워드 정보가 주파수 도메인 오디오 표현(152)을 표현한다.

[0100] 오디오 인코더(100)는 또한 산술 코드워드 정보(172a)를 수신하도록 구성되는 비트스트림 페이로드 포맷터(bitstream payload formatter, 190)를 포함한다. 비트스트림 페이로드 포맷터(190)는, 또한 일반적으로, 예를 들어, 어떤 스케일링 인자들이 스케일러/양자화기(150)에 의해 적용되었는지를 기술하는 스케일링 인자 정보와 같은 추가 정보를 수신하도록 구성된다. 또한, 비트스트림 페이로드 포맷터(190)는 다른 제어 정보를 수신하도록 구성될 수 있다. 비트스트림 페이로드 포맷터(190)는, 하기에서 논의될 것으로, 요구되는 비트스트림 구문에 따라 비트스트림을 모아서 수신된 정보에 기초해 비트스트림(112)을 제공하도록 구성된다.

[0101] 다음에서, 산술 인코더(170)에 관한 세부사항들이 기술될 것이다. 산술 인코더(170)는 주파수 도메인 오디오 표현(132)에 대한 복수의 후처리되고 스케일링되며 양자화된 스펙트럼 값들을 수신하도록 구성된다. 산술 인코더는, 스펙트럼 값으로부터, 또는 심지어 두 개의 스펙트럼 값들로부터, 최상위 비트 평면 m을 추출하도록 구성되는 최상위비트 평면 추출기(174)를 포함한다. 여기서 최상위 비트 평면은 스펙트럼 값의 최상위 비트들인 하나 또는 심지어 그 이상의 비트들(예를 들어, 2 내지 3 비트)을 포함할 수 있음을 알아야 한다. 그러므로, 최상위 비트 평면 추출기(174)는 스펙트럼 값의 최상위 비트 평면 값(176)을 제공한다.

[0102] 그렇지 않으면, 그러나, 최상위 비트 평면 추출기(174)는 복수의 스펙트럼 값들(예를 들어, 스펙트럼 값들 a 및 b)의 최상위 비트 평면들을 결합하는 결합된 최상위 비트 평면 값 m을 제공할 수 있다. 스펙트럼 값(a)의 최상위 비트 평면은 m으로 지칭된다. 그렇지 않으면, 복수의 스펙트럼 값들 a, b의 결합된 최상위 비트 평면 값이 m으로 지칭된다.

[0103] 산술 인코더(170)는 또한 최상위 비트 평면 값 m을 표현하는 산술 코드워드(acod_m[pki][m])를 결정하도록 구성되는 제1 코드워드 결정기(180)를 포함한다. 선택적으로, 코드워드 결정기(180)는 또한, 예를 들어, 얼마나 많은 하위 비트 평면들이 이용 가능한지를 가리키는(그리고, 결과적으로, 최상위 비트 평면의 수치적 가중치(weight)를 가리키는) (여기서 "ARITH_ESCAPE"라고도 지칭되는) 하나 이상의 이스케이프(escape) 코드워드들을 제공할 수 있다. 제1 코드워드 결정기(180)는 누적 빈도 테이블 인덱스(pki)를 갖는(또는 누적 빈도 테이블 인덱스(pki)에 의해 참조되는) 선택된 누적 빈도 테이블을 이용하여 최상위 비트 평면 값 m과 연관된 코드워드를 제공하도록 구성될 수 있다.

[0104] 어떤 누적 빈도 테이블이 선택되어야 하는지에 관해 결정하기 위해, 산술 인코더는, 바람직하게는, 예를 들어, 어떤 스펙트럼 값들이 이전에 인코딩되었는지를 관찰하여, 산술 인코더의 상태를 추적하도록 구성되는 상태 추적기(182)를 포함한다. 상태 추적기(182)는 결과적으로, 상태 정보(184), 예를 들어, "s", 또는 "t", 또는 "c"로 지칭되는 상태 값을 제공한다. 산술 인코더(170)는 또한 상태 정보(184)를 수신하고, 코드워드 결정기(180)에 선택된 누적 빈도 테이블을 기술하는 정보(188)를 제공하도록 구성되는 누적 빈도 테이블 선택기(186)를 포함한다. 예를 들어, 누적 빈도 테이블 선택기(186)는 96개의 누적 빈도 테이블들의 셋트 중에서 어떤 누적 빈도 테이블이 코드워드 결정기에 의해 사용되기 위해 선택되는지 기술하는 누적 빈도 테이블 인덱스 "pki"를 제공할

수 있다. 그렇지 않으면, 누적 빈도 테이블 선택기(186)는 코드워드 결정기에 선택된 누적 빈도 테이블 전체 또는 서브 테이블을 제공할 수 있다. 그러므로, 코드워드 결정기(180)는 최상위 비트 평면 값 m의 코드워드 acod_m[pki][m]의 제공을 위해 선택된 누적 빈도 테이블 또는 서브 테이블을 이용할 수 있어, 최상위 비트 평면 값 m을 인코딩하는 실제 코드워드 acod_m[pki][m]가 m 값 및 누적 빈도 테이블 인덱스 pki에 따르게 되고, 결과적으로 현재 상태 정보(184)에 따르게 된다. 코딩 과정 및 획득된 코드워드 포맷에 관한 더 상세한 설명들이 하기에서 기술될 것이다.

[0105] 그러나, 몇몇 실시예들에서, 상태 추적기 182는 상태 추적기 750, 상태 추적기 1050, 또는 상태 추적기 1250과 동일하거나, 상태 추적기 750, 상태 추적기 1050, 또는 상태 추적기 1250의 기능을 가질 수 있음을 알아야 한다. 누적 빈도 테이블 선택기 186은, 몇몇 실시예들에서, 맵핑 규칙 선택기 760, 맵핑 규칙 선택기 1060, 또는 맵핑 규칙 선택기 1260과 동일하거나, 맵핑 규칙 선택기 760, 맵핑 규칙 선택기 1060, 또는 맵핑 규칙 선택기 1260의 기능을 가질 수 있음을 또한 알아야 한다. 또한, 제1 코드워드 결정기 180는, 몇몇 실시예들에서, 스펙트럼 값 인코딩(740)과 동일하거나 스펙트럼 값 인코딩(740)의 기능을 가질 수 있다.

[0106] 산술 인코더(170)는, 만약 인코딩되는 하나 이상의 스펙트럼 값들이 오직 최상위 비트 평면만을 이용하여 인코딩 가능한 값들의 범위를 초과하면, 스케일링되고 양자회된 주파수 도메인 오디오 표현(152)으로부터 하나 이상의 하위 비트 평면들을 추출하도록 구성되는 하위 비트 평면 추출기(189a)를 추가로 포함한다. 하위 비트 평면들은, 요구에 따라, 하나 이상의 비트을 포함할 수 있다. 이에 따라, 하위 비트 평면 추출기(189a)는 하위 비트 평면 정보(189b)를 제공한다. 산술 인코더(170)는 또한, 하위 비트 평면 정보(189d)를 수신하고, 그에 기초하여, 0개, 1개, 또는 그 이상의 하위 비트 평면들의 콘텐츠를 표현하는 0개, 1개 또는 그 이상의 코드워들 "acord_r"을 제공하도록 구성되는 제2 코드워드 결정기(189c)를 포함한다. 제2 코드워드 결정기(189c)는 하위 비트 평면 정보(189b)로부터 하위 비트 평면 코드워드들 "acod_r"를 도출하기 위해 산술 인코딩 알고리즘 또는 임의의 다른 인코딩 알고리즘을 적용하도록 구성될 수 있다.

[0107] 여기서 하위 비트 평면들의 수는 스케일링되고 양자화된 스펙트럼 값들(152)의 값에 따라 변할 수 있어, 만약 인코딩되는 스케일링되고 양자화된 스펙트럼 값이 비교적 작으면 하위 비트 평면이 전혀 없을 수 있으며, 인코딩되는 현재 스케일링되고 양자화된 스펙트럼 값이 중간 범위이면 하나의 하위 비트 평면이 있을 수 있고, 인코딩되는 스케일링되고 양자화된 스펙트럼 값이 비교적 큰 값을 취하면 하나 이상의 하위 비트 평면이 있을 수 있다는 것을 알아야 한다.

[0108] 상기를 요약하면, 산술 인코더(170)는, 계층적 인코딩 과정을 이용하여, 정보(152)에 의해 기술되는 스케일링되고 양자화된 스펙트럼 값들을 인코딩하도록 구성된다. 하나 이상의 스펙트럼 값들의 (예를 들어, 스펙트럼 값마다 1, 2, 또는 3 비트를 포함하는) 최상위 비트 평면은 최상위 비트 평면 값 m의 산술 코드워드 "acod_m[pki][m]"를 획득하기 위해 인코딩된다. 하나 이상의 스펙트럼 값들의 하나 이상의 하위 비트 평면들(예를 들어, 1, 2, 또는 3 비트를 포함하는 각각의 하위 비트 평면들)은 하나 이상의 코드워드들 "acod_r"을 획득하기 위해 인코딩된다. 최상위 비트 평면을 인코딩할 때, 최상위 비트 평면의 값 m은 코드워드 acod_m[pki][m]에 맵핑된다. 이를 위해, 96 개의 각각 다른 누적 빈도 테이블들이 산술 인코더(170)의 상태에 따라, 즉 이전에 인코딩된 스펙트럼 값들에 따라 값 m의 인코딩을 위해 이용 가능하다. 이에 따라, 코드워드 "acod_m[pki][m]"가 획득된다. 또한, 만약 하나 이상의 하위 비트 평면들이 존재한다면, 하나 이상의 코드워드들 "acod_r"이 비스트스림에 제공되어 비트스트림에 포함된다.

[0109] 재설정 설명

[0110] 오디오 인코더(100)는, 선택적으로, 콘텍스트를 재설정하여, 예를 들어, 디폴트 값에 상태 인덱스를 설정하여 비트율의 개선이 얻어낼 수 있는지 여부를 결정하도록 구성될 수 있다. 이에 따라, 오디오 인코더(100)는 산술 인코딩을 위한 콘텍스트가 재설정되었는지 여부를 가리키고, 또한 상응하는 디코더에서 산술 디코딩을 위한 콘

텍스트가 재설정되어야 하는지 여부를 가리키는 (예를 들어, "arith_reset_flag"라고 불리는) 재설정 정보를 제공하도록 구성될 수 있다.

[0111] 비트스트림 형식 및 적용된 누적 빈도 테이블들에 관한 세부사항들이 하기에서 논의될 것이다.

[0112] 9. 도 2에 따른 오디오 디코더

[0113] 다음에서는, 본 발명의 일 실시예에 따른 오디오 디코더가 기술될 것이다. 도 2는 그러한 오디오 디코더(200)에 대한 블록 도식도를 도시한다.

[0114] 오디오 디코더 200는 인코딩된 오디오 정보를 표현하고 오디오 인코더 100에 의해 제공된 비트스트림 112과 동일할 수 있는 비트스트림 210을 수신하도록 구성된다. 오디오 디코더(200)는 비트스트림(210)에 기초하여 디코딩된 오디오 정보(212)를 제공한다.

[0115] 오디오 디코더(200)는 비트스트림(210)을 수신하여, 비트스트림(210)으로부터 인코딩된 주파수 도메인 오디오 표현(222)을 추출하도록 구성되는 선택적 비트스트림 페이로드 디 포맷터(220)를 포함한다. 비트스트림 페이로드 디 포맷터(220)는, 비트스트림(210)으로부터, 예를 들어, 스펙트럼 값(a), 또는 복수의 스펙트럼 값들 a, b의 최상위 비트 평면 값 m을 표현하는 산술 코드워드 "acod_m[pki][m]", 및 주파수 도메인 오디오 표현의 스펙트럼 값(a), 또는 복수의 스펙트럼 값들 a, b의 하위 비트 평면의 콘텐츠를 표현하는 코드워드 "acod_r"와 같은 산술적으로 코딩된 스펙트럼 데이터를 추출하도록 구성될 수 있다. 그러므로, 인코딩된 주파수 도메인 오디오 표현(222)는 스펙트럼 값의 산술적으로 인코딩된 표현을 이룬다(또는 포함한다). 비트스트림 페이로드 디포맷터(220)는, 도 2에 도시되지 않은, 추가 제어 정보를 비트스트림으로부터 추출하도록 추가로 구성된다. 또한, 비트스트림 페이로드 디포맷터는, 선택적으로, 산술 재설정 플래그 또는 "arith_reset_flag"로도 지칭되는 상태 재설정 정보(224)를 비트스트림(210)으로부터 추출하도록 구성된다.

[0116] 오디오 디코더(200)는 "스펙트럼 무잡음 디코더"라고도 지칭되는 산술 디코더(230)를 포함한다. 산술 디코더(230)는 인코딩된 주파수 도메인 오디오 표현(200), 및, 선택적으로, 상태 재설정 정보(224)를 수신하도록 구성된다. 산술 디코더(230)는 또한, 스펙트럼 값들의 디코딩된 표현을 포함할 수 있는 디코딩된 주파수 도메인 오디오 표현(232)을 제공하도록 구성된다. 예를 들어, 디코딩된 주파수 도메인 오디오 표현(232)은 인코딩된 주파수 도메인 오디오 표현(220)에 의해 기술되는 스펙트럼 값들의 디코딩된 표현을 포함할 수 있다.

[0117] 오디오 디코더(200)는 또한, 디코딩된 주파수 도메인 오디오 표현(232)를 수신하고, 그에 기초하여, 역 양자화되고 재스케일링된 주파수 도메인 오디오 표현(242)을 제공하도록 구성되는 선택적 역 양자화기/재스케일러(240)를 포함한다.

[0118] 오디오 디코더(200)는 역 양자화되고 재스케일링된 주파수 도메인 오디오 표현(242)을 수신하고, 그에 기초하여, 역 양자화되고 재스케일링된 주파수 도메인 오디오 표현(242)의 전처리된 버전(252)을 제공하도록 구성되는 선택적 스펙트럼 전처리기(250)를 추가로 포함한다. 오디오 디코더(200)는 "신호 변환기"라고도 지칭되는 주파수 도메인 대 시간 도메인 신호 전환기(260)를 또한 포함한다. 신호 전환기(260)는 역 양자화되고 재스케일링된 주파수 도메인 오디오 표현(242)의 전처리된 버전(252)(또는, 그렇지 않으면, 역 양자화되고 재스케일링된 주파수 도메인 오디오 표현(242) 또는 디코딩된 주파수 도메인 오디오 표현(232))을 수신하고, 그에 기초하여, 오디오 정보의 시간 도메인 표현(262)을 제공하도록 구성된다. 주파수 도메인 대 시간 도메인 신호 전환기(260)는, 예를 들어, 역 변형 이산 코사인 변환(inverse-modified-discrete-consine-transform, IMDCT) 및 (예를 들어, 중첩 및 추가와 같은 다른 보조 기능들뿐만 아니라) 적절한 윈도윙을 수행하기 위한 전환기를 포함

한다.

[0119] 오디오 디코더(200)는 오디오 정보의 시간 도메인 표현(262)을 수신하고, 시간 도메인 후처리를 이용하여 디코딩된 오디오 정보(212)를 획득하도록 구성되는 선택적 시간 도메인 후처리기(270)를 추가로 포함할 수 있다. 그러나, 만약 후처리가 생략된다면, 시간 도메인 표현(262)은 디코딩된 오디오 정보(212)와 동일할 수 있다.

[0120] 역 양자화기/재스케일러(240), 스펙트럼 전처리기(250), 주파수 도메인 대 시간 도메인 신호 전환기(260), 및 시간 도메인 후처리기(270)는 비트스트림 페이로드 디포맷터(220)에 의해 비트스트림(210)으로부터 추출되는 제어 정보에 따라 제어될 수 있음을 여기서 알아야 한다.

[0121] 오디오 디코더(200)의 전반적인 기능을 요약하면, 디코딩된 주파수 도메인 오디오 표현(232), 예를 들어, 인코딩된 오디오 정보의 오디오 프레임과 연관된 스펙트럼 값들의 셋트는 산술 디코더(230)를 이용하여 인코딩된 주파수 도메인 표현(222)에 기초해 획득될 수 있다. 이어서, MDCT 계수들일 수 있는, 예를 들어, 1024 개의 스펙트럼 값들의 셋트가 역 양자화되며, 재스케일링되고, 전처리된다. 이에 따라, 스펙트럼 값들(예를 들어, 1024개의 MDCT 계수들)의 역 양자화되며, 재스케일링되고, 스펙트럼 전처리된 셋트가 획득된다. 그 이후에, 오디오 프레임의 시간 도메인 표현이 주파수 도메인 값들(예를 들어, MDCT 계수들)의 역 양자화되며, 재스케일링되고, 스펙트럼 전처리된 셋트로부터 도출된다. 이에 따라, 오디오 프레임의 시간 도메인 표현이 획득된다. 소정의 오디오 프레임의 시간 도메인 표현은 이전 및/또는 이어지는 오디오 프레임들의 시간 도메인 표현들과 결합될 수 있다. 예를 들어, 이어지는 오디오 프레임들의 시간 도메인 표현들 사이의 중첩 및 추가는 인접한 오디오 프레임들의 시간 도메인 표현들 사이의 과도(transition)를 평활화하고(smoothen), 에일리어싱 무효화(aliasing cancellation)를 획득하기 위해 수행될 수 있다. 디코딩된 시간 주파수 도메인 오디오 표현(232)에 기초한 디코딩된 오디오 정보(212)의 복원에 관한 세부사항들에 대하여, 예를 들어, 세부적인 논의가 주어지는, 국제 표준 ISO/IEC 14496-3 제3장 제4절이 참조된다. 그러나, 다른 좀더 정교한 중첩 및 에일리어싱 무효화 기법들이 이용될 수 있다.

[0122] 다음에서, 산술 디코더(230)에 관한 몇몇 세부사항들이 기술될 것이다. 산술 디코더(230)는 최상위 비트 평면 값 m을 기술하는 산술 코드워드(acod_m[pki][m])를 수신하도록 구성되는 최상위 비트 평면 결정기(284)를 포함한다. 최상위 비트 평면 결정기(284)는 산술 코드워드 "acod_m[pki][m]"로부터 최상위 비트 평면 값 m을 도출하기 위해 복수의 96개의 누적 빈도 테이블들 포함하는 셋트 중에서 하나의 누적 빈도 테이블을 이용하도록 구성될 수 있다.

[0123] 최상위 비트 평면 결정기(284)는 코드워드(acod_m)에 기초하여 하나 이상의 스펙트럼 값들의 최상위 비트 평면의 값들(286)을 도출하도록 구성된다. 산술 디코더(230)는 스펙트럼 값의 하나 이상의 하위 비트 평면들을 표현하는 하나 이상의 코드워드들 "acod_r"을 수신하도록 구성되는 하위 비트 평면 결정기(288)를 추가로 포함한다. 이에 따라, 하위 비트 평면 결정기(288)는 하나 이상의 하위 비트 평면들의 디코딩된 값들(290)을 제공하도록 구성된다. 오디오 디코더(200)는, 만약 그러한 하위 비트 평면들이 현재 스펙트럼 값들에 대해 이용가능하다면, 하나 이상의 스펙트럼 값들의 최상위 비트 평면의 디코딩된 값들(286) 및 스펙트럼 값들의 하나 이상의 하위 비트 평면들의 디코딩된 값들(290)을 수신하도록 구성되는 비트 평면 결합기(292)를 또한 포함한다. 이에 따라, 비트 평면 결합기(292)는 디코딩된 주파수 도메인 오디오 표현(232)의 일부분인 디코딩된 스펙트럼 값들을 제공한다. 당연히, 산술 디코더(230)는 일반적으로, 오디오 콘텐츠의 현재 프레임과 연관된 디코딩된 스펙트럼 값들의 전체 셋트를 획득하기 위해 복수의 스펙트럼 값들을 제공하도록 구성된다.

[0124] 산술 디코더(230)는 산술 디코더의 상태를 기술하는 상태 인덱스(298)에 따라 96개의 누적 빈도 테이블들 중 하나를 선택하도록 구성되는 누적 빈도 테이블 선택기(296)를 추가로 포함한다. 산술 디코더(230)는 이전에 디코딩된 스펙트럼 값들에 따라 산술 디코더의 상태를 추적하도록 구성되는 상태 추적기(299)를 추가로 포함한다. 상태 정보는, 선택적으로, 상태 재설정 정보(224)에 응답하여 디폴트 상태 정보로 재설정될 수 있다. 이에

따라, 누적 빈도 테이블 선택기(296)는, 코드워드 "acod_m"에 따라 최상위 비트 평면 값 m의 디코딩에서 적용하기 위해, 선택된 누적 빈도 테이블, 또는 선택된 누적 빈도 테이블, 또는 그 자체의 서브 테이블의 인덱스(예를 들어, pki)을 제공하도록 구성된다.

[0125] 오디오 디코더(200)의 기능을 요약하면, 오디오 디코더(200)는 비트율 효율적으로 인코딩된 주파수 도메인 오디오 표현(222)을 수신하고, 그에 기초하여 디코딩된 주파수 도메인 오디오 표현을 획득하도록 구성된다. 인코딩된 주파수 도메인 오디오 표현(222)에 기초하여 디코딩된 주파수 도메인 오디오 표현(232)을 획득하기 위해 이용되는 산술 디코더(230)에서, 인접한 스펙트럼 값들의 최상위 비트 평면의 값들의 각각 다른 결합들에 대한 확률이 누적 빈도 테이블을 적용하도록 구성되는 산술 디코더(280)를 이용하여 활용된다. 다시 말해서, 스펙트럼 값들 사이의 통계적 의존성이 이전에 계산된 디코딩된 스펙트럼 값들을 관찰하여 획득되는 상태 인덱스(298)에 따라 96 개의 각각 다른 누적 빈도 테이블들을 포함하는 셋트 중에서 각각 다른 누적 빈도 테이블들을 선택함으로써 활용된다.

[0126] 상태 추적기(299)는 상태 추적기 826, 상태 추적기 1126, 또는 상태 추적기 1326과 동일할 수 있거나, 상태 추적기 826, 상태 추적기 1126, 또는 상태 추적기 1326의 기능을 가질 수 있음을 알아야 한다. 누적 빈도 테이블 선택기(296)는 맵핑 규칙 선택기 828, 맵핑 규칙 선택기 1128, 또는 맵핑 규칙 선택기 1328과 동일할 수 있거나, 맵핑 규칙 선택기 828, 맵핑 규칙 선택기 1128, 또는 맵핑 규칙 선택기 1328의 기능을 가질 수 있다. 최상위 비트 평면 결정기(284)는 스펙트럼 값 결정기(824)와 동일할 수 있거나, 스펙트럼 값 결정기(824)의 기능을 가질 수 있다.

[0127] 10. 스펙트럼 무잡음 코딩의 수단에 대한 개요

[0128] 다음에서, 예를 들어, 산술 인코더(170) 및 산술 디코더(230)에 의해 수행되는 인코딩 및 디코딩 알고리즘에 관한 세부사항들이 설명될 것이다.

[0129] 디코딩 알고리즘의 설명에 초점을 둔다. 그러나, 상응하는 인코딩 알고리즘이 디코딩 알고리즘의 사상(teaching)에 따라 수행될 수 있음을 알아야 하는데, 여기서 인코딩되고 디코딩된 스펙트럼 값들 사이의 맵핑들은 역으로 되고, 여기서 맵핑 규칙 인덱스 값들에 대한 계산은 실질적으로 동일하다. 인코더에서, 인코딩된 스펙트럼 값들은 디코딩된 스펙트럼 값들의 자리를 대체한다. 또한, 인코딩되는 스펙트럼 값들은 디코딩되는 스펙트럼 값들의 자리를 대체한다.

[0130] 다음에서 논의될 디코딩은 일반적으로 후처리되며, 스케일링되고, 양자화된 스펙트럼 값들의 이른바 "스펙트럼 무잡음 코딩"을 가능하게 하기 위해 이용된다는 것을 알아야 한다. 스펙트럼 무잡음 코딩은, 예를 들어, 에너지 압축 시간 도메인 대 주파수 도메인 전환기에 의해 획득되는 양자화된 스펙트럼의 중복을 더 줄이기 위해 오디오 인코딩/디코딩 구상(또는 임의의 다른 인코딩/디코딩 구상)에서 이용된다. 본 발명의 실시예들에서 이용되는 스펙트럼 무잡음 코딩 기법은 동적으로 적응된 콘텍스트과 함께 산술 코딩에 기초한다.

[0131] 본 발명에 따른 몇몇 실시예들에서, 스펙트럼 무잡음 코딩 기법은 2-튜플들(tuples)에 기초하는데, 즉, 두 개의 근처에 있는(neighbored) 스펙트럼 계수들이 결합된다. 각각의 2-튜플은 부호, 최상위 2 비트 방식 평면, 및 잔여 하위 비트 평면들로 나누어진다. 최상위 2 비트 방식 평면 m에 대한 무잡음 코딩은 4 개의 이전에 디코딩된 2-튜플로부터 도출된 콘텍스트에 따르는 누적 빈도 테이블들을 이용한다. 무잡음 코딩은 양자화된 스펙트럼 값들에 의해 공급되고, 4 개의 이전에 디코딩된 근처에 있는 2-튜플들로부터 도출된 콘텍스트에 따르는 누적 빈도 테이블들을 이용한다. 여기서, 도 4에 도시된 바와 같이, 시간과 주파수 모두에서 근처에 있는 것으로 고려된다. (하기에서 설명될 것으로) 누적 빈도 테이블들은, 그 다음에, 가변 길이 이진 코드를 발생시키기 위해 산술 코더에 의해 (그리고 가변 길이 이진 코드로부터 디코딩된 값들을 도출하기 위해 산술 디코더에 의해)

이용된다.

[0132] 예를 들어, 산술 디코더(170)는 심볼들 및 그 각각의 확률의(즉, 각각의 확률에 따르는) 소정의 셋트에 대한 이진 코드를 만들어낸다. 이진 코드는 코드워드에, 심볼들의 셋트가 있는 확률 구간을 맵핑함으로써 발생된다.

[0133] 잔여 하위 비트 평면 r의 무잡음 코딩은 단일 누적 빈도 테이블을 이용한다. 누적 빈도는, 예를 들어, 하위 비트 평면들에서 일어나는 심볼들의 균일한 분포에 상응하는데, 즉, 하위 비트 평면들에서 0 또 1이 일어나는 확률이 동일한 것으로 예상된다.

[0134] 다음에서, 스펙트럼 무잡음 코딩의 수단에 대한 다른 간단한 개요가 제공될 것이다. 스펙트럼 무잡음 코딩은 양자화된 스펙트럼의 중복을 더 줄이는데 이용된다. 스펙트럼 무잡음 코딩 기법은 동적으로 적응된 콘텍스트과 함께 산술 코딩에 기초한다. 무잡음 코딩은 양자화된 스펙트럼 값들에 의해 공급되고, 예를 들어, 스펙트럼 값들의 4 개의 이전에 디코딩된 근처에 있는 2-튜플들로부터 도출된 콘텍스트에 따르는 누적 빈도 테이블들을 이용한다. 여기서, 근처에 있는 것은, 도 4에 도시된 바와 같이, 시간 및 주파수 모두에서인 것으로 고려된다. 누적 빈도 테이블들은, 그 다음에, 가변 길이 이진 코드를 발생시키기 위해 산술 코더에 의해 이용된다.

[0135] 산술 코더는 심볼들 및 그 각각의 확률의 소정의 셋트에 대한 이진 코드를 만들어낸다. 이진 코드는, 코드워드에, 심볼들의 셋트가 있는 확률 구간을 맵핑하여 발생된다.

[0136] 11. 디코딩 과정

[0137] 11.1 디코딩 과정 개요

[0138] 다음에서, 복수의 스펙트럼 값들을 디코딩하는 과정에 대한 의사 프로그램 코드 표현을 도시하는 도 3을 참조하여, 스펙트럼 값 코딩의 과정에 대한 개요가 주어질 것이다.

[0139] 복수의 스펙트럼 값들의 디코딩 과정은 콘텍스트의 초기화(310)를 포함한다. 콘텍스트의 초기화(310)는, 함수 "arith_map_context(N, arith_reset)flag)"를 이용하여, 이전 콘텍스트으로부터 현재 콘텍스트의 도출을 포함한다. 이전 콘텍스트으로부터의 현재 콘텍스트 도출은 콘텍스트의 재설정을 선택적으로 포함한다. 콘텍스트의 재설정 및 이전 콘텍스트으로부터의 현재 콘텍스트 도출 모두가 하기에서 논의될 것이다.

[0140] 복수의 스펙트럼 값들의 디코딩은 스펙트럼 값 디코딩(312), 및 하기에서 기술되는 함수 "arith_update_context(i,a,b)"에 의해 수행되는 콘텍스트 업데이트(313)의 반복을 또한 포함한다. 이른바 "ARITH_STOP" 심볼이 감지되지 않는 한, 스펙트럼 값 디코딩(312) 및 콘텍스트 업데이트(312)는 lg/2 회 반복되는데,여기서 lg/2는 (예를 들어, 오디오 프레임에 대한) 디코딩되는 스펙트럼 값들의 2-튜플들의 수를 가리킨다. 또한, lg 스펙트럼 값들의 셋트의 디코딩은 부호 디코딩(314) 및 종료 단계(315)를 또한 포함한다.

[0141] 스펙트럼 값들의 튜플의 디코딩(312)은 콘텍스트 값 계산(312a), 최상위 비트 평면 디코딩(312b), 산술 중지 심볼 감지(312c), 하위 비트 평면 추가(312d), 및 어레이 업데이트(312e)를 포함한다.

[0142] 상태 값 계산(312a)은, 예를 들어, 도 5c 또는 5d에 도시된 바와 같이, 함수 "arith_get_context(c,i,N)"의 호출을 포함한다. 이에 따라, 수치적 현재 콘텍스트 (상태) 값 c이 함수 "arith_get_context(c,i,N)"의 함수 호출에 대한 반환 값으로써 제공된다. 알 수 있는 바와 같이, 함수 "arith_get_context(c,i,N)"에 입력 변수로 쓰일

수 있는 ("c"로도 지칭되는) 수치적 이전 콘텍스트 값은, 수치적 현재 콘텍스트 값 c을 반환 값으로 획득하기 위해 업데이트된다.

[0143] 최상위 비트 평면 디코딩(312b)는 디코딩 알고리즘(312ba), 및 상기 알고리즘(312ba)의 결과 값 mm으로부터의 값들 a, b 도출(312bb)의 반복적인 실행을 포함한다. 알고리즘(312ba)의 준비에서 변수 lev는 0으로 초기화된다. "break" 명령(또는 조건(condition))에 도달할 때까지 알고리즘(312ba)은 반복된다. 알고리즘(312ba)은, 수치적 현재 콘텍스트 값 c에 따라, 또한 하기에서 (그리고, 예를 들어, 도 5e 및 5f에서 도시된 실시예들에서) 논의되는 함수 "arith_get_pk()"를 이용하여 레벨 값 "esc_nb"에 따라 (누적 빈도 테이블 인덱스로도 쓰일 수 있는) 상태 인덱스 "pki"의 계산을 포함한다. 알고리즘(312ba)은 함수 "arith_get_pk"의 호출에 의해 반환되는 상태 인덱스 "pki"에 따르는 누적 빈도 테이블의 선택을 또한 포함하는데, 여기서 변수 "cum_freq"는 상태 인덱스 "pki"에 따라 96 개의 누적 빈도 테이블들 (또는 서브 테이블들) 중에서 하나의 시작 주소에 설정될 수 있다. 변수 "cfl"은, 또한, 예를 들어, 알파벳에서 심볼들의 수, 즉, 디코딩될 수 있는 각각 다른 값들의 수와 동일한 선택된 누적 빈도 테이블(또는 서브 테이블)의 길이로 초기화될 수 있다. 16 개의 각각 다른 최상위 비트 평면 값들 및 하나의 이스케이프 심볼("ARITH_ESCAPE")이 디코딩될 수 있기 때문에, 최상위 비트 평면 값 m의 디코딩을 위해 이용 가능한 "ari_cf_m[pki=0][17]"에서부터 "ari_cf_m[pki=95][17]"까지의 모든 누적 빈도 테이블들(또는 서브 테이블들)의 길이는 17이다.

[0144] 이어서, 최상위 비트 평면 값 m은, (변수 "cum_freq" 및 변수 "cfl"에 의해 기술된) 선택된 누적 빈도 테이블을 고려해, 함수 "arith_decode()"를 실행하여 획득될 수 있다. 최상위 비트 값 m을 도출할 때, 비트스트림(210) 중에 "acod_m"이라고 불리는 비트들이 평가될 수 있다(예를 들어, 도 6g 또는 도 6h 참고).

[0145] 알고리즘(312ba)은 최상위 비트 평면 값 m이 이스케이프 심볼 "ARITH_ESCAPE"과 같은지 아닌지 여부를 검사하는 것을 또한 포함한다. 만약 최상위 비트 평면 값 m이 산술적 이스케이프 심볼과 같지 않다면, 알고리즘(312ba)이 중단되고("break" 조건), 알고리즘(312ba)의 남은 명령들은, 그러면, 건너뛰게 된다. 이에 따라, 312bb 단계에서 값 b 및 값 a를 셋팅하여 처리과정(process)의 실행이 계속된다. 그에 반해서, 디코딩된 최상위 비트 값 m이 산술적 이스케이프 심볼, 또는 "ARITH_ESCAPE"과 같다면, 레벨 값 "lev"이 1만큼 증가한다. 변수 "lev"가 7보다 크지 않으면, 레벨 값 "esc_nb"는 레벨 값 "lev"와 같게 설정되는데, 이 경우에, 변수 "esc_nb"는 7과 같게 설정된다. 언급한 바와 같이, 알고리즘(312ba)은, 그 다음에, 디코딩된 최상위 비트 평면 값 m이 반복 산술 이스케이프 심볼과 다를 때까지 반복되는데, 여기서, (함수 "arith_get_pk()"의 입력된 파라미터가 변수 "esc_nb"의 값에 따라 적응되기 때문에) 수정된 콘텍스트가 이용된다.

[0146] 알고리즘(312ba)의 1회 실행 또는 반복 실행을 이용하여 최상위 비트 평면이 디코딩되자마자, 즉, 산술 이스케이프 심볼과 다른 최상위 비트 평면 값 m이 디코딩되면, 스펙트럼 값 변수 "b"는 최상위 비트 평면 값 m의 복수의 (예를 들어, 2) 좀더 유효한 비트들과 같게 설정되고, 스펙트럼 값 변수 "a"는 최상위 비트 값 m의 (예를 들어, 2) 최하위 비트로 설정된다. 이러한 기능에 관한 세부사항들은, 예를 들어, 도면 부호 312bb에서 알 수 있다.

[0147] 이어서, 산술 중지 심볼이 존재하는지 여부가 312c 단계에서 검사된다. 이는 최상위 비트 평면 값 m이 0과 같고, 변수 "lev"가 0보다 큰 경우이다. 이에 따라, 최상위 비트 평면 값 m이 0과 같은 "비정상" 조건에 의해 산술 중지 조건이 신호로 알려지는 한편, 변수 "lev"는 증가된 수치적 가중치가 최상위 비트 평면 값 m에 연관된다고 가리킨다. 다시 말해, 만약 비트스트림이, 정상적인 인코딩의 경우에서는 일어나지 않는 조건인, 최소 수치적 가중치보다 높은 증가된 수치적 가중치가 0과 같은 최상위 비트 평면 값으로 주어져야 한다고 가리키면, 산술 중지 조건이 감지된다. 다시 말해, 만약 인코딩된 산술적 이스케이프 심볼에 이어 인코딩된 최상위 비트 평면 값 0이 뒤따른다면, 산술 중지 조건이 신호로 알려진다.

[0148] 312c 단계에서 수행되는, 산술 중지 조건이 있는지 여부에 대한 평가 이후에, 예를 들어, 도 3에서 도면 부호

312d로 도시된 바와 같이, 하위 비트 평면들이 획득된다. 각각의 하위 비트 평면에 대해, 2 개의 이진 값들이 디코딩된다.

[0149] 이진 값들 중 하나는 변수 a(또는 스펙트럼 값들의 튜플의 제1 스펙트럼 값)와 연관되고, 이진 값들 중 하나는 변수 b(또는 스펙트럼 값들의 튜플의 제2 스펙트럼 값)와 연관된다. 하위 비트 평면들의 수는 변수 lev로 지칭된다.

[0150] (만약에 있다면) 하나 이상의 하위 비트 평면들에 대한 디코딩에서 알고리즘(212da)이 반복적으로 수행되는데, 여기서 알고리즘(212da)의 실행 횟수는 변수 "lev"에 의해 결정된다. 여기서 알고리즘(212ba)의 제1 반복은 212bb 단계에서 설정된 바와 같이 변수들 a, b의 값들에 기초하여 수행됨을 알아야 한다. 알고리즘(212da)의 추가적 반복들은 변수 a, b의 업데이트된 변수 값들에 기초하여 수행된다.

[0151] 반복의 시작에서, 누적 빈도 테이블이 선택된다. 이어서, 변수 r의 값을 획득하기 위해 산술 디코딩이 수행되는데, 여기서 변수 r의 값은 복수의 하위 비트들, 예를 들어, 변수 a와 연관된 하나의 하위 비트 및 변수 b와 연관된 하나의 하위 비트를 기술한다. 함수 "ARITH_DECODE"는 변수 r을 획득하기 위해 이용되는데, 여기서 누적 빈도 테이블 "arith_cf_r"이 산술 디코딩에 이용된다.

[0152] 이어서, 변수들 a 및 b의 값들이 업데이트된다. 이를 위해, 변수 a는 1 비트 만큼 왼쪽으로 이동되고, 이동된 변수 a의 최하위 비트는 변수 r의 최하위 비트에 의해 정의된 값으로 설정된다. 변수 b는 1 비트 만큼 왼쪽으로 이동되고, 이동된 변수 b의 최하위 비트는 변수 r의 비트 1에 의해 정의된 값으로 설정되는데, 여기서 변수 r의 비트 1은 변수 r의 이진 표현에서 수치적 가중치 2를 갖는다. 알고리즘 412ba가, 그 다음에, 모든 최하위 비트들이 디코딩 될때까지 반복된다.

[0153] 하위 비트 평면들에 대한 디코딩 이후에, 어레이 인덱스 2*i 및 2*i+1을 갖는 어레이의 엔트리들에 변수들 a, b의 값들이 저장되어 있는 어레이 "x_ac_dec"이 업데이트된다.

[0154] 이어서, 함수 "arith_update_context(i,a,b)"를 호출하여 콘텍스트 상태가 업데이트되는데, 그에 대한 세부사항들이 도 5g를 참조하여 하기에서 설명될 것이다.

[0155] 313 단계에서 수행되는 콘텍스트 상태의 업데이트에 이어, 연속 변수(running variable) i가 값 lg/2에 도달하거나 산술 중지 조건이 감지될 때까지, 알고리즘들 312 및 313이 반복된다.

[0156] 이어서, 도면 부호 315에서 알 수 있는 바와 같이, 종료 알고리즘 "arith_finish()"이 수행된다. 종료 알고리즘 "arith_finish()"에 대한 세부사항들이 도 5m을 참조하여 하기에서 기술될 것이다.

[0157] 종료 알고리즘 315에 이어, 알고리즘 314를 이용하여 스펙트럼 값들의 부호들이 디코딩된다. 알 수 있는 바와 같이, 0과 다른 스펙트럼 값들의 부호들은 개별적으로 코딩된다. 알고리즘 314에서, 0이 아닌 i=0과 i=lg-1 사이의 인덱스들 i를 갖는 모든 스펙트럼 값들에 대한 부호들이 판독된다. i=0과 i=lg-1 사이의 스펙트럼 값 인덱스 i를 갖는 각각의 0이 아닌 스펙트럼 값들에 대해, 값(일반적으로 단일 비트) s가 비트스트림으로부터 판독된다. 만약 비트스트림으로부터 판독되는 s 값이 1과 같다면, 상기 스펙트럼 값의 부호가 도치된다. 이를 위해, 인덱스 i를 갖는 스펙트럼 값이 0과 같은지 여부 결정 및 디코딩된 스펙트럼 값들의 부호 업데이트 모두를 위하여, 어레이 "x_ac_dec"에 대한 접근이 이루어진다. 그러나, 변수들 a, b에 대한 부호들은 부호 디코딩(314)에서 달라지지 않은 채로 남아 있음을 알아야 한다.

[0158] 부호 디코딩(314) 이전에 종료 알고리즘(315)을 수행하여, ARITH_STOP 심볼 후의 모든 필요한 빈들(bins)을 재설정할 수 있다.

[0159] 여기서 하위 비트 평면들의 값들의 획득에 대한 구상은 본 발명에 따른 몇몇 실시예들에 특별히 관련되는 것은 아님을 알아야 한다. 몇몇 실시예들에서, 임의의 하위 비트 평면들에 대한 디코딩은 심지어 생략될 수도 있다. 그렇지 않으면, 각각 다른 디코딩 알고리즘들이 이를 위해 이용될 수 있다.

[0160] 11.2 도 4에 따른 디코딩 순서

[0161] 다음에서는, 스펙트럼 값들에 대한 디코딩 순서가 기술될 것이다.

[0162] 양자화된 스펙트럼 계수들 "x_ac_dec[]"는 가장 낮은 주파수 계수에서 시작하여 가장 높은 주파수 계수로 나아가며 무잡음 인코딩되어 (예를 들어, 비트스트림으로) 전송된다.

[0163] 결과적으로, 양자화된 스펙트럼 계수들 "x_ac_dec[]"는 가장 낮은 주파수 계수에서 시작하여 가장 높은 주파수 계수로 나아가며 무잡음 디코딩된다. 양자화된 스펙트럼 계수들은 이른바 ({a,b}라고도 지칭되는) 2-튜플(a,b)로 모이는 2개의 연속적인(예를 들어, 주파수에서 인접하는) 계수들 a 및 b의 집합들로 디코딩된다. 여기서 양자화된 스펙트럼 계수들은 종종 "qdec"라고도 지칭됨을 알아야 한다.

[0164] 주파수 도메인 모드를 위한 디코딩된 계수들 "x_ac_dec[]"(ISO/IEC 14496 제3장 제4절에서 논의된 바와 같이, 예를 들어 변형 이산 코사인 변환을 이용하여 획득된, 예를 들어, 고급 오디오 코딩을 위한 디코딩된 계수들)이, 그 다음에, 어레이 "x_ac_quant[g][win][sfb][bin]"에 저장된다. 무잡음 코딩 코드워드들의 전송 순서는 그것들이 어레이에 수신되어 저장된 순서로 디코딩될 때, "bin"은 가장 빠르게 증가하는 인덱스이고, "g"는 가장 느리게 증가하는 인덱스이다. 코드워드 내에서, 디코딩 순서는 a, b이다.

[0165] 변환 코딩 여기(transform coded-excitation, TCX)을 위한 디코딩된 계수들 "x_ac_dec[]"는, 예를 들어, 어레이 "x_tcx_invquant[win][bin]"에 바로 저장되고, 무잡음 코딩 코드워드의 전송 순서는 그것들이 어레이에 수신되어 저장되는 순서로 디코딩될 때, "bin"은 가장 빠르게 증가하는 인덱스이고, "win"은 가장 느리게 증가하는 인덱스이다. 코드워드 내에서, 디코딩 순서는 a, b이다. 다시 말해, 만약 스펙트럼 값들이 음성 코더의 선형 예측 필터의 변환 코딩 여기를 기술하면, 스펙트럼 값들 a, b는 변환 코딩 여기의 인접한 증가하는 주파수들에 연관된다. 낮은 주파수에 연관된 계수들은 높은 주파수에 연관된 계수보다 일반적으로 먼저 인코딩되고 디코딩된다.

[0166] 특히, 오디오 디코더(200)는, 주파수 도메인 대 시간 도메인 신호 전환을 이용하는 시간 도메인 오디오 신호 표현의 "직접적" 발생, 및 주파수 도메인 대 시간 도메인 신호 전환기의 출력에 의해 활성화되는 주파수 도메인 대 시간 도메인 디코더와 선형 예측 필터를 모두 이용하는 시간 도메인 오디오 신호 표현의 "간접적" 제공 모두를 위해, 산술 디코더(230)에 의해 제공되는 디코딩된 주파수 도메인 표현(232)을 적용하도록 구성될 수 있다.

[0167] 다시 말해, 산술 디코더는, 여기서 상세히 논의되는 그 기능이, 주파수 도메인에서 인코딩된 오디오 콘텐츠의 시간 주파수 도메인 표현의 스펙트럼 값들의 디코딩, 및 선형 예측 도메인에서 인코딩된 음성 신호를 디코딩(또는 합성)하도록 적응된 선형 예측 필터에 대한 자극 신호의 시간 주파수 도메인 표현 제공에 매우 적합하다. 그러므로, 산술 디코더는 주파수 도메인 인코딩된 오디오 콘텐츠 및 선형 예측 주파수 도메인 인코딩된 오디오 콘

텐츠(변형 코딩 여기 선형 예측 도메인 모드)를 모두 다룰 수 있는 오디오 디코더에 사용하는데 매우 적합하다.

[0168] 11.3 도 5a 및 5b에 따른 콘텍스트 초기화

[0169] 다음에서는, 310 단계에서 수행되는 ("콘텍스트 맵핑"이라고도 지칭되는) 콘텍스트 초기화가 기술될 것이다.

[0170] 콘텍스트 초기화는, 그 제1 예시가 도 5a에 도시되고 그 제2 예시가 도 5b에 도시되는, 알고리즘 "arith_map_context()"에 따른 과거 콘텍스트과 현재 콘텍스트 사이의 맵핑를 포함한다.

[0171] 알 수 있는 바와 같이, 현재 콘텍스트는, 제1 크기(dimension) 2 및 제2 크기 "n_context"를 갖는 어레이의 형태를 취하는 전역 변수(global variable) "q[2][n_context]"에 저장된다. 과거 콘텍스트는 (이용된다면) "n_context"의 크기를 갖는 테이블의 형태를 취하는 변수 "qs[n_context]"에 선택적으로 (그러나 반드시 그렇지는 않음) 저장될 수 있다.

[0172] 도 5a에서의 예시 알고리즘 "arith_map_context"를 참조하면, 입력 변수 N은 현재 윈도우의 길이를 기술하고, 입력 변수 "arith_reset_flag"는 콘텍스트가 재설정되어야 하는지 여부를 가리킨다. 또한, 전역 변수 "previous_N"는 이전 윈도우 길이를 기술한다. 여기서, 일반적으로 윈도우와 연관된 스펙트럼 값들의 수치는, 적어도 대략, 시간 도메인 샘플들의 측면에서 상기 윈도우 길이의 반(half)과 같다는 것을 알아야 한다. 또한, 스펙트럼 값들의 2-튜플의 길이의 수치는, 결과적으로, 적어도 대략, 시간 도메인 샘플들의 측면에서 상기 윈도우 길이의 1/4과 같다는 것을 알아야 한다.

[0173] 도 5a의 예시를 참조하면, 콘텍스트의 맵핑은 알고리즘 "arith_map_context()"에 따라 수행될 수 있다. 여기서, 만약 플래그 "arith_reset_flag"가 활성화되어 결과적으로 콘텍스트가 재설정되어야 한다고 가리키면, 함수 "arith_map_context()"는 j=0 내지 j=N/4-1에 대하여 현재 콘텍스트 어레이 q의 엔트리들 "q[0][j]"을 0으로 설정함을 알아야 한다. 그렇지 않으면, 즉, "arith_reset_flag"가 비활성화되면, 현재 콘텍스트 어레이 q의 엔트리들 "q[0]0[j]"은 현재 콘텍스트 어레이 q의 엔트리들 "q-[1][k]"로부터 도출된다. 현재 (예를 들어, 주파수 도메인 인코딩된) 오디오 프레임과 연관된 스펙트럼 값들의 수가 j=k=0 내지 j=k=N/4-1에 대해 이전 오디오 프레임과 연관된 스펙트럼 값들의 수와 같다면, 도 5a에 따른 함수 "arith_map_context()"는 현재 콘텍스트 어레이 q의 엔트리들 "q[0][j]"을 현재 콘텍스트 어레이 q의 값들 "q[1][k]"로 설정한다.

[0174] 현재 오디오 프레임에 연관된 스펙트럼 값들의 수가 이전 오디오 프레임에 연관된 스펙트럼 값들의 수와 다르다면 더 복잡한 맵핑이 수행된다. 그러나, 이 경우에 맵핑에 관한 세부사항들은 본 발명의 핵심 발상에 특별히 관계되지 않아, 세부사항들을 위해 도 5a의 의사 프로그램 코드가 참고된다.

[0175] 또한, 수치적 현재 콘텍스트 값 c에 대한 초기화 값은 함수 "arith_map_context()"에 의해 반환된다. 이 초기화 값은, 예를 들어, 12 비트 만큼 왼쪽으로 이동된 엔트리 "q[0][0]"의 값과 같다. 이에 따라, 수치적 (현재) 콘텍스트 값 c은 반복적 업데이트를 위해 적절히 초기화된다.

[0176] 또한, 도 5b는 대안적으로 이용될 수 있는 알고리즘 "arith_map_context()"에 대한 다른 예시를 도시한다. 세부사항들을 위해, 도 5b에서 의사 프로그램 코드가 참조된다.

[0177] 상기를 요약하면, 플래그 "arith_reset_flag"는 콘텍스트가 재설정되어야 하는지를 결정한다. 만약 플래그가 참

(true)이라면, 알고리즘 "arith_map_context()"의 재설정 서브 알고리즘(500a)이 호출된다. 그렇지 않으면, 그러나, 만약 플래그 "arith_reset_flag"가 (콘텍스트에 대한 어떠한 재설정되 수행되지 말아야 함을 가리키는) 비활성화이면, 디코딩 과정은, 콘텍스트 성분 벡터(또는 어레이) q가 q[1][] 내지 q[9][]에 저장된 이전 프레임의 콘텍스트 성분들을 복사하고 맵핑함으로써 업데이트되는 초기화 단계부터 시작한다. q 내의 콘텍스트 성분들은 2-튜플 당 4 비트로 저장된다. 콘텍스트 성분의 복사 및/또는 맵핑은 서브 알고리즘(500b)에서 수행된다.

[0178] 도 5b의 예시에서, 디코딩 과정은 qs에 저장된 보관된 과거 콘텍스트 및 현재 프레임 의 콘텍스트 q 사이의 맵핑이 이루어지는 초기화 단계부터 시작한다. 과거 콘텍스트 qs는 주파수 라인당 2 비트로 저장된다.

[0179] 11.4 도 5c 및 5d에 따른 상태 값 계산

[0180] 다음에서는, 상태 값 계산(312a)이 좀더 상세히 기술될 것이다.

[0181] 제1 예시적 알고리즘은 도 5c를 참조하여 기술될 것이고, 제2 예시적 알고리즘은 도 5d를 참조하여 기술될 것이다.

[0182] (도 3에 도시된 바와 같이) 수치적 현재 콘텍스트 값 c는, 그 의사 프로그램 코드 표현이 도 5c에 도시되는 함수 "arith_get_context(c,i,N)"의 반환 값으로 획득될 수 있음을 알아야 한다. 그렇지 않으면, 그러나, 수치적 현재 콘텍스트 값 c는, 그 의사 프로그램 코드 표현이 도 5d에 도시되는 함수 "arith_get_context(c,i)"의 반환 값으로 획득될 수 있다.

[0183] 상태 값의 계산과 관련하여, 상태 평가, 즉, 수치적 현재 콘텍스트 값 c의 계산에 이용되는 콘텍스트를 도시하는 도 4가 또한 참조된다. 도 4는 시간 및 주파수 모두에 대한 스펙트럼 값들의 2차원 표현을 도시한다. 가로 좌표(410)는 시간을 기술하고 세로 좌표(412)는 주파수를 기술한다. 도 4에서 알 수 있는 바와 같이, (바람직하게는 수치적 현재 콘텍스트 값을 이용하여) 디코딩하기 위한 스펙트럼 값들의 튜플(420)은 시간 인덱스 t0 및 주파수 인덱스 i와 연관된다. 알 수 있는 바와 같이, 시간 인덱스 t0에 대하여, 주파수 인덱스들 i-1, i-2, 및 i-3을 갖는 튜플들은, 주파수 인덱스 i를 갖는 튜플(120)의 스펙트럼 값들이 디코딩되는 시점에서 이미 디코딩된다. 도 4로부터 알 수 있는 바와 같이, 시간 인덱스 t0 및 주파수 인덱스 i-1을 갖는 스펙트럼 값 430은 스펙트럼 값들의 튜플 420이 디코딩되기 전에 이미 디코딩되고, 스펙트럼 값들의 튜플 430은 스펙트럼 값들의 튜플 420의 디코딩에 이용되는 콘텍스트를 위해 고려된다. 유사하게, 시간 인덱스 t0-1 및 주파수 인덱스 i-1을 갖는 스펙트럼 값들의 튜플 440, 시간 인덱스 t0-1 및 주파수 인덱스 i를 갖는 스펙트럼 값들의 튜플 450, 및 시간 인덱스 t0-1 및 주파수 인덱스 i+1을 갖는 스펙트럼 값들의 튜플 460은, 스펙트럼값들의 튜플 420이 디코딩 되기에 전에 이미 디코딩되고, 스펙트럼 값들의 튜플 420을 디코딩하는데 이용되는 콘텍스트의 결정을 위해 고려된다. 튜플 420의 스펙트럼 값들이 디코딩되는 시점에서 이미 디코딩되고 콘텍스트를 위해 고려된 스펙트럼 값들(계수들)이 음영이 들어간 정사각형으로 도시된다. 그에 반해서, (튜플 420의 스펙트럼 값들이 디코딩되는 시점에서) 이미 디코딩되었으나 (튜플 420의 스펙트럼 값들의 디코딩을 위한) 콘텍스트를 위해 고려되지 않는 몇몇 다른 스펙트럼 값들은 대시 기호로된 선들을 갖는 정사각형들로 표현되고, (튜플 420의 스펙트럼 값들이 디코딩되는 시점에서 아직 디코딩되지 않은) 다른 스펙트럼 값들은 대시 기호로 된 선을 갖는 원들로 도시된다. 대시 기호로 된 선들을 갖는 정사각형들로 표현된 튜플들 및 대시 기호로 된 선들을 갖는 원들로 표현된 튜플들은 튜플 420의 스펙트럼 값들을 디코딩을 위한 콘텍스트를 결정하는데 이용되지 않는다.

[0184] 그러나, 튜플 420의 스펙트럼 값들을 디코딩을 위한 콘텍스트의 "보통의" 또는 "정상적인" 계산에 이용되지 않는 이러한 스펙트럼 값들 중 몇몇은, 그럼에도 불구하고, 개별적으로 또는 함께 그 크기와 관련하여 미리 결정된 조건을 만족시키는 복수의 이전에 디코딩된 인접한 스펙트럼 값들의 감지를 위해 평가될 수 있다는 것을 알아야 한다. 이 사안에 관한 세부사항들이 하기에서 논의될 것이다.

[0185] 이제 도 5c를 참조하면, 알고리즘 "arith_get_context(c,i,N)"에 대한 세부사항들이 기술될 것이다. 도 5c는 종래의 잘 알려진 C 언어 및/또는 C++ 언어를 이용하는 의사 프로그램 코드의 형태로 함수 "arith_get_context(c,i,N)"의 기능을 도시한다. 그러므로, 함수 "arith_get_context(c,i,N)"에 의해 수행되는 수치적 현재 콘텍스트 값 "c"의 계산에 관한 몇몇 좀더 세부사항들이 기술될 것이다.

[0186] 함수 "arith_get_context(c,i,N)"는, 입력 변수들로써, 수치적 이전 콘텍스트 값 c에 의해 기술될 수 있는 "과거 상태 콘텍스트(old state context)"를 수신한다는 것을 알아야 한다. 함수 "arith_get_context(c,i,N)"는 또한, 입력 변수로써, 디코딩하기 위한 스펙트럼 값들의 2-튜플의 인덱스 i를 수신한다. 인덱스 i는 일반적으로 주파수 인덱스이다. 입력 변수 N은 스펙트럼 값들이 디코딩되기 위한 윈도우의 윈도우 길이를 기술한다.

[0187] 함수 "arith_get_context(c,i,N)", 출력 값으로써, 업데이트된 상태 콘텍스트를 기술하고 수치적 현재 콘텍스트 값으로 고려될 수 있는 입력 변수 c의 업데이트된 버전을 제공한다. 요약하면, 함수 "arith_get_context(c,i,N)"는 입력 변수로써 수치적 이전 콘텍스트 값 c을 수신하고, 수치적 현재 콘텍스트 값으로 고려되는 그것의 업데이트된 버전을 제공한다. 또한, 함수 "arith_get_context(c,i,N)"는 변수들 i, N을 고려하고, 또한 "전역" 어레이 q[][]에 접근한다.

[0188] 함수 "arith_get_context(c,i,N)"의 세부사항들과 관련하여, 처음에 이진 형태로 수치적 이전 콘텍스트 값을 표현하는 변수 c는 504a 단계에서 4 비트 만큼 오른쪽으로 이동됨을 알아야 한다. 이에 따라, (입력 변수 c에 의해 표현된) 수치적 이전 콘텍스트 값의 4 개의 최하위 4 비트가 버려진다. 또한, 수치적 이전 콘텍스트 값들의 다른 비트에 대한 수치적 가중치들이, 예를 들어, 16 개의 인자로 감소된다.

[0189] 또한, 만약 2-튜플의 인덱스 i가 N/4-1 보다 작으면, 즉, 최대 값을 취하지 않으면, 엔트리 q[0][i+1]의 값이 504a 단계에서 획득되는 이동된 콘텍스트 값의 비트 12 내지 15(즉, 2^{12}, 2^{13}, 2^{14}, 및 2^{15}의 수치적 가중치를 갖는 비트)에 추가되므로, 수치적 현재 콘텍스트 값이 수정된다. 이를 위해, 어레이 q[][]의 엔트리 q[0][i+1(또는, 좀더 정확하게는, 상기 엔트리에 의해 표현된 값의 이진 표현)는 12 비트 만큼 왼쪽으로 이동된다. 엔트리 q[0][i+1에 의해 표현된 값의 이동된 버전은, 그 다음에, 504a 단계에서 도출되는 콘텍스트 값 c, 즉, 수치적 이전 콘텍스트 값의 비트 이동된(4 비트 만큼 오른쪽으로 이동된) 수치 표현에 추가된다. 여기서 어레이 q[][]의 엔트리 q[0][i+1는 오디오 콘텐츠의 이전 부분(예를 들어, 도 4를 참조하여 정의된, 시간 인덱스 t0-1을 갖는 오디오 콘텐츠의 일부분), 및 (함수 "arith_get_context(c,i,N)"에 의해 출력된 수치적 현재 콘텍스트 값 c을 이용하여) 현재 디코딩되는 스펙트럼 값들의 튜플보다 더 높은 주파수(예를 들어, 도 4를 참조하여 정의된 것과 같이, 주파수 인덱스 i+1를 갖는 주파수)와 연관된 서브구역 값을 표현한다는 것을 알아야 한다. 다시 말해, 스펙트럼 값들의 튜플 420이 수치적 현재 콘텍스트 값을 이용하여 디코딩된다면, 엔트리 q[0][i+1]는 이전에 디코딩된 스펙트럼 값들의 튜플 460에 기초할 수 있다.

[0190] (12 비트 만큼 왼쪽으로 이동된) 어레이 q[][]의 엔트리 q[0][i+1]에 대한 선택적 추가가 도면 부호 504b에 도시된다. 알 수 있는 바와 같이, 엔트리 q[0][i+1]에 의해 표현된 값의 추가는, 당연히 오직, 주파수 인덱스 i가 가장 높은 주파수 인덱스 i=N/4-1를 갖는 스펙트럼 값들의 튜플을 지칭하지 않을 경우에만 수행된다.

[0191] 이어서, 504c 단계에서, 변수 c에 대한 업데이트된 값을 획득하기 위해 변수 c의 값이 16진 값 0xFFF0과 AND 결합되는 부울 AND 연산이 수행된다. 그러한 AND 연산을 수행하여, 변수 c의 4 개의 최하위 비트들이 효과적으로 0으로 설정될 수 있다.

[0192] 504d 단계에서, 엔트리 q[1][i-1]의 값은 504c 단계에서 획득되는 변수 c의 값에 추가되어, 변수 c의 값을 업데

이트한다. 그러나, 504c 단계에서의 변수 c의 상기 업데이트는 오직 디코딩하기 위한 2-튜플의 주파수 인덱스 i가 0보다 더 큰 경우에만 수행된다. 엔트리 q[1][i-1]는 수치적 현재 콘텍스트 값을 이용하여 디코딩되는 스펙트럼 값들의 주파수들보다 더 작은 주파수들에 대하여 오디오 콘텐츠의 현재 부분에 대한 이전에 디코딩된 스펙트럼 값들의 튜플에 기초하는 콘텍스트 서브구역 값임을 알아야 한다. 스펙트럼 값들의 튜플 420이 함수 "arith_get_context(c,i,N)"의 현재의 실행에 의해 반환된 수치적 현재 콘텍스트 값을 이용하여 디코딩된다고 추정되면,예를 들어, 어레이 q[][]의 엔트리 q[1][i-1]는 시간 인덱스 t0 및 주파수 인덱스 i-1을 갖는 튜플 430과 연관될 수 있다.

[0193] 요약하면, 수치적 이전 콘텍스트 값의 비트 0, 1, 2, 및 3(즉, 4 개의 최하위 비트 부분)는 수치적 이전 콘텍스트 값의 이진 수치적 표현 밖으로 그것들을 이동시켜 504a 단계에서 버려진다. 또한, 이동된 변수 c(즉, 이동된 수치적 이전 콘텍스트 값)의 비트 12, 13, 14, 및 15는 504b 단계에서 콘텍스트 서브구역 값 q[0][i+1]에 의해 정의된 값들을 취하도록 설정된다. 이동된 수치적 이전 콘텍스트 값의 비트 0, 1, 2, 및 3(즉, 원래의 수치적 이전 콘텍스트 값의 비트 4, 5, 6, 및 7)이 504c 및 504d 단계에서 콘텍스트 서브구역 값 q[1][i-1]으로 덮어씌어진다.

[0194] 결과적으로, 수치적 이전 콘텍스트 값의 비트 0 내지 3은 스펙트럼 값들의 튜플 432와 연관된 콘텍스트 서브구역 값을 표현하며, 수치적 이전 콘텍스트 값의 비트 4 내지 7은 이전에 디코딩된 스펙트럼 값들의 튜플 434와 연관된 콘텍스트 서브구역 값을 표현하며, 수치적 이전 콘텍스트 값의 비트 8 내지 11은 이전에 디코딩된 스펙트럼 값들의 튜플 440과 연관된 콘텍스트 서브구역 값을 표현하고, 수치적 이전 콘텍스트 값의 비트 12 내지 15는 이전에 디코딩된 스펙트럼 값들의 튜플 450과 연관된 콘텍스트 서브구역 값을 표현한다고 할 수 있다. 함수 "arith_get_context(c,i,N)"로 입력되는 수치적 이전 콘텍스트 값은 스펙트럼 값들의 튜플 430의 디코딩과 연관된다.

[0195] 함수 "arith_get_context(c,i,N)"의 출력 변수로써 획득되는 수치적 현재 콘텍스트 값은 스펙트럼 값들의 튜플 420의 디코딩과 연관된다. 이에 따라, 수치적 현재 콘텍스트 값들의 비트 0 내지 3은 스펙트럼 값들의 튜플 430과 연관된 콘텍스트 서브구역 값을 기술하며, 수치적 현재 콘텍스트 값의 비트 4 내지 7은 스펙트럼 값들의 튜플 440과 연관된 콘텍스트 서브구역 값을 기술하며, 수치적 현재 콘텍스트 값의 비트 8 내지 11은 스펙트럼 값의 튜플 450과 연관된 수치적 서브구역 값을 기술하고, 수치적 현재 콘텍스트 값의 비트 12 내지 15는 스펙트럼 값들의 튜플 460과 연관된 콘텍스트 서브구역 값을 기술한다. 그러므로, 수치적 이전 콘텍스트 값의 부분, 즉, 수치적 이전 콘텍스트 값의 비트 8 내지 15는, 수치적 현재 콘텍스트 값의 비트 4 내지 11과 같이, 수치적 현재 콘텍스트 값에 또한 포함된다는 것을 알 수 있다. 그에 반해서, 현재 수치적 이전 콘텍스트 값의 비트 0 내지 7은 수치적 이전 콘텍스트 값의 수치 표현으로부터 수치적 현재 콘텍스트 값의 수치적 표현을 도출할 때 버려진다.

[0196] 504e 단계에서, 만약 디코딩하기 위한 2-튜플의 주파수 인덱스 i가 미리 결정된 수, 예를 들어, 3보다 더 크면, 수치적 현재 콘텍스트 값을 표현하는 변수 c는 선택적으로 업데이트 된다. 이 경우에, 즉, 만약 i가 3보다 크면, 콘텍스트 서브구역 값들 q[1][i-3], q[1][i-2], 및 q[1][i-1]의 합이 미리 결정된 값, 예를 들어, 5보다 더 작은지(또는 같은지) 여부가 결정된다. 만약 상기 콘텍스트 서브구역 값들의 합이 상기 미리 결정된 값보다 더 작다고 확인되면, 16진 값, 예를 들어, 0x10000이 변수 c에 추가된다. 이에 따라, 변수 c는, 만약 콘텍스트 서브구역 값들 q[1][i-3], q[1][i-2], 및 q[1][i-1]이 특히 작은 합 값을 포함하한다는 조건이 있는지를 변수 c가 가리키도록 설정된다. 예를 들어, 수치적 현재 콘텍스트 값의 비트 16은 그러한 조건을 가리키기 위해 플래그로 작용할 수 있다.

[0197] 결론적으로 말하면, 함수 "arith_get_context(c,i,N)"의 반환 값은 504a, 504b, 504c, 504d, 및 504e 단계에 의해 결정되며, 수치적 현재 콘텍스트 값은 504a, 504b, 504c, 및 504d 단계에서 수치적 이전 콘텍스트 값으로부터 도출되고, 여기서, 대체로, 특히 작은 절대 값들을 갖는 이전에 디코딩된 스펙트럼 값들의 환경을 가리키는 플래그는 504e 단계에서 도출되어 변수 c에 추가된다. 이에 따라, 만약 504e 단계에서 평가된 조건이 만족되

지 않는다면, 504a, 504b, 504c, 504d 단계에서 획득된 변수 c의 값이, 504f 단계에서, 함수 "arith_get_context(c,i,N)"의 반환 값으로써, 반환된다. 그에 반해서, 504e 단계에서 평가된 조건이 만족된다면, 504a, 504b, 504c, 및 504d 단계에서 도출되는 변수 c의 값은 16진 값 0x10000 만큼 증가되고, 504e 단계에서 이 증가 연산의 결과가 반환된다.

[0198] 상기를 요약하면, (하기에서 좀더 상세히 기술될 것으로) 무잡음 디코더는 무부호 양자화된 스펙트럼 계수들의 2-튜플들을 출력한다는 것을 알아야 한다. 처음에, 콘텍스트의 상태 c가 디코딩하기 위한 2-튜플들에 "관련되는(surrounding)" 이전에 디코딩된 스펙트럼 계수들에 기초하여 계산된다. 일 바람직한 실시예에서, (예를 들어, 수치적 콘텍스트 값에 의해 표현되는) 상태는, 오직 2개의 새로운 2-튜플들(예를 들어 2-튜플들 430 및 460)을 고려하여, (수치적 이전 콘텍스트 값으로 지칭되는) 마지막 디코딩된 2-튜플의 콘텍스트 상태를 이용하여 증가하여 업데이트된다. 상기 상태는 (예를 들어, 수치적 현재 콘텍스트 값의 수치 표현을 이용하여) 17 비트로 코딩되고, 함수 "arith_get_context()"에 의해 반환된다. 세부적인 사항들을 위해, 도 5c의 프로그램 코드 표현이 참조된다.

[0199] 또한, 함수 "arith_get_context()"의 대안적인 실시예에 대한 의사 프로그램 코드가 도 5d에 도시됨을 알아야 한다. 도 5d에 따른 함수 "arith_get_context(c,i)"는 도 5c에 따른 함수 "arith_get_context(c,i,N)"와 유사하다. 그러나, 도 5d에 따른 함수 "arith_get_context(c,i)"는 최소 주파수 인덱스 i=0 또는 최대 주파수 인덱스 i=N/4-1을 포함하는 스펙트럼 값들의 튜플들에 대한 특별한 처리나 디코딩을 포함하지 않는다.

[0200] 11.5 맵핑 규칙 선택

[0201] 다음에서는, 맵핑 규칙의 선택, 예를 들어, 심볼 코드로의 코드워드 값의 맵핑을 기술하는 누적 빈도 테이블이 기술될 것이다. 맵핑 규칙의 선택은 수치적 현재 콘텍스트 값 c에 의해 기술되는 콘텍스트 상태에 따라 이루어진다.

[0202] 11.5.1 도 5e에 따른 알고리즘을 이용하는 맵핑 규칙 선택

[0203] 다음에서는, 함수 "arith_get_pk(c)"를 이용하는 맵핑 규칙의 선택이 기술될 것이다. 스펙트럼 값들의 튜플을 제공하기 위해 코드 값 "acod_m"을 디코딩할 때, 함수 "arith_get_pk()"가 서브 알고리즘(312b)의 시작에서 호출된다는 것을 알아야 한다. 알고리즘(312b)의 각각 다른 반복에서 각각 다른 인수들(arguments)을 갖는 함수 "arith_get_pk(c)"가 호출된다는 것을 알아야 한다. 예를 들어, 알고리즘(312b)의 제1 반복에서, 312a 단계에서 함수 "arith_get_context(c,i,N)"의 이전 실행에 의해 제공된 수치적 현재 상태 값 c와 같은 인수를 갖는 함수 "arith_get_pk(c)"이 호출된다. 그에 반해서, 서브 알고리즘(312ba)의 추가 반복들에서, 312a 단계에서 함수 "arith_get_context(c,i,N)"에 의해 제공된 수치적 현재 콘텍스트 값 c의 합, 및 변수 "esc_nb"의 값의 비트 이동된 버전인 인수를 갖는 함수 "arith_get_pk(c)"가 호출되는데, 여기서 변수 "esc_nb"의 값은 17 비트 만큼 왼쪽으로 이동된다. 그러므로, 함수 "arith_get_context(c,i,N)"에 의해 제공된 수치적 현재 콘텍스트 값 c이 알고리즘(312ba)의 제1 반복, 즉, 비교적 작은 스펙트럼 값들의 디코딩에서 함수 "arith_get_pk()"의 입력 값으로 이용된다. 그에 반해서, 비교적 큰 스펙트럼 값들을 디코딩할 때, 도 3에 도시된 바와 같이, 변수 "esc_nb"의 값이 고려되어, 함수 "arith_get_pk()"의 입력 변수가 수정된다.

[0204] 이제, 함수 "arith_get_pk(c)"의 제1 실시예에 대한 의사 프로그램 코드 표현을 도시하는 도 5e를 참조하면, 함수 "arith_get_pk()"가 입력 값으로써 변수 c를 수신함을 알아야 하는데, 여기서 변수 c는 콘텍스트의 상태를 기술하고, 여기서 함수 "arith_get_pk()"의 입력 변수 c는 적어도 몇몇 상황들에서 함수 "arith_get_context()"에 의해 반환 변수로 제공된 수치적 현재 콘텍스트 값과 같다. 또한, 함수 "arith_get_pk()"는, 출력 변수로써, 확률 모델의 인덱스를 기술하고 맵핑 규칙 인덱스 값으로 고려될 수 있는

변수 "pki"를 제공한다는 것을 알아야 한다.

[0205] 도 5e를 참조하면, 함수 "arith_get_pk()"가 변수 초기화(506a)를 포함한다는 것을 알 수 있는데, 여기서 변수 "i_min"는 -1의 값을 취하도록 초기화된다. 유사하게, 변수 i가 변수 "i_min"와 같게 설정되어, 변수 i도 -1의 값으로 초기화된다. 변수 "i_max"는 테이블 "ari_lookup_m[]"의 엔트리들의 수보다 1 만큼 작은 값을 취하도록 초기화된다(이에 대한 세부사항들은 도 21a 및 21b를 참조하여 기술될 것이다). 이에 따라, 변수들 "i_min" 및 "i_max"가 구간을 정의한다.

[0206] 이어서, 테이블 "ari_hash_m"의 엔트리를 지칭하는 인덱스 값을 식별하기 위해 검색(506b)이 수행되어, 함수 "arith_get_pk()"의 입력 변수 c의 값이 상기 엔트리와 인접한 엔트리에 의해 정의된 구간 내에 있게 된다.

[0207] 검색(506b)에서, 서브 알고리즘(506ba)이 반복되는 한편, 변수들 "i_max"와 "i_min" 사이의 차이는 1보다 더 크다. 서브 알고리즘(506ba)에서, 변수 i는 변수들 "i_min"과 "i_max"의 값들의 산술 평균과 같게 설정된다. 결과적으로, 변수 i는 변수들 "i_min"와 "i_max"의 값들에 의해 정의된 테이블 구간의 중간에서 테이블 "ari_hash_m[]"의 엔트리를 지칭한다. 이어서, 변수 j는 테이블 "ari_hash_m[]"의 엔트리 "ari_hash_m[i]"의 값과 같게 설정된다. 그러므로, 변수 j는 변수들 "i_min"과 "i_max"에 의해 정의된 테이블 구간의 중간에 엔트리가 있는 테이블 "ari_hash_m[]"의 엔트리에 의해 정의된 값을 취한다. 이어서, 만약 함수 "arith_get_pk()"의 입력 변수 c의 값이 테이블 "ari_hash_m[]"의 테이블 엔트리 "j=ari_hash_m[i]"의 최상위 비트들에 의해 정의된 상태 값과 다르면, 변수들 "i_min"과 "i_max"에 의해 정의된 구간이 업데이트된다. 예를 들어, 테이블 "ari_hash_m[]"의 엔트리들의 "상위 비트"(비트 8 및 그 위쪽(bits 8 and upward))는 유효 상태 값을 기술한다. 이에 따라, 값 "j>>8"는 해시 테이블 인덱스 값 i에 의해 지칭된 테이블 "ari_hash_m[]"의 엔트리 "j=ari_hash_m[i]"에 의해 표현된 유효 상태 값을 기술한다. 이에 따라, 만약 변수 c의 값이 값 "j>>8"보다 더 작다면, 이는 변수 c에 의해 기술된 상태 값이 테이블 "ari_hash_m[]의 엔트리 "ari_hash_m[i]"에 의해 기술된 유효 상태 값보다 더 작다는 것을 의미한다. 이 경우에, 변수 "i_max"의 값이 변수 i의 값과 같게 설정되는데, 이는 결국 "i_min" 및 "i_max"에 의해 정의된 구간의 크기가 감소되는 효과를 갖는데, 여기서 새로운 구간은 이전 구간의 하부쪽 반과 거의 동일하다. 만약, 변수 c에 의해 기술된 콘텍스트 값이 어레이 "ari_hash_m[]"에 의 엔트리 "ari_hash_m[i]"에 의해 기술된 유효 상태 값보다 더 크다는 의미인, 함수 "arith_get_pk()"의 입력 변수 c가 값 "j>>8"보다 더 크다는 것이 확인된다면, 변수 "i_min"의 값은 변수 i의 값과 같게 설정된다. 이에 따라, 변수들 "i_min" 및 "i_max"의 값들에 의해 정의된 구간의 크기는 변수들 "i_min" 및 "i_max"의 이전 값들에 의해 정의된 이전 구간의 크기의 대략 반으로 감소된다. 좀더 정확하게, 변수 c의 값이 엔트리 "ari_hash_m[i]"에 의해 정의된 유효 상태 값보다 더 큰 경우, 변수 "i_min"의 업데이트 된 값 및 변수 "i_max"의 이전(달라지지 않은) 값에 의해 정의된 구간은 이전 구간의 상부쪽 반과 거의 같다.

[0208] 만약, 그러나, 알고리즘 "arith_get_pk()"의 입력 변수 c에 의해 기술된 콘텍스트 값이 엔트리 "ari_hash_m[i]"에 의해 정의된 유효 상태 값과 같다고 확인되면 (즉, c==(j>>8)), 엔트리 "ari_hash_m[i]"의 최하위 8 비트에 의해 정의된 맵핑 규칙 인덱스 값은 함수 "arith_get_pk()"의 반환 값으로써 반환된다(명령어 "return(j&0xFF)").

[0209] 상기를 요약하면, 그 최상위 비트(비트 8 및 그 위쪽)이 유효 상태 값을 기술하는 엔트리 "ari_hash_m[i]"는 각각의 반복(506ba)에서 평가되고, 함수 "arith_get_pk()"의 입력 변수 c에 의해 기술된 콘텍스트 값(또는 수치적 현재 콘텍스트 값)은 테이블 엔트리 "ari_hash_m[i]"에 의해 기술된 유효 상태 값과 비교된다. 만약 입력 변수 c에 의해 표현된 콘텍스트 값이 테이블 엔트리 "ari_hash_m[i]"에 의해 표현된 유효 상태 값보다 더 작다면, 테이블 구간의 (변수 "i_max"로 기술된) 상부 경계가 감소되고, 만약 입력 변수 c에 의해 기술된 콘텍스트 값이 테이블 엔트리 "ari_hash_m[i]"에 의해 기술된 유효 상태 값보다 더 크면, 테이블 구간의 (변수 "i_min"의 값으로 기술되는) 하부 경계가 증가된다. 상기 두 경우들 모두에서, ("i_max"와 "i_min" 사이의 차이에 의해 정의된) 구간의 크기가 1 보다 작거나, 1과 같지 않으면, 서브 알고리즘(506ba)이 반복된다. 만약, 반대로, 변수 c에 의해 기술된 콘텍스트 값이 테이블 엔트리 "ari_hash_m[i]"에 의해 기술된 유효 상태 값과 같다면, 함

수 "arith_get_pk()"은 중단되는데, 여기서 반환 값은 테이블 엔트리 "ari_hash_m[i]"의 최하위 8 비트에 의해 정의된다.

[0210] 만약, 그러나, 구간 크기가 그 최소 값("i_max" - "i_min"이 1보다 작거나, 1과 같음)에 도달하여 검색(506b)이 종료되면, 함수 "arith_get_pk()"의 반환 값은 테이블 "ari_lookup_m[]"의 엔트리 "ari_lookup_m[i_max]"에 의해 결정되는데, 이는 도면 부호 506c에서 알 수 있다. 이에 따라, 테이블 "ari_hash_m[]"의 엔트리들은 유효 상태 값들 및 구간들의 경계들을 모두를 정의한다. 서브 알고리즘(506ba)에서, 검색 구간 경계들 "i_min" 및 "i_max"가 반복적으로 적응되어, 그 해시 테이블 인덱스 i가 적어도 대략적으로 구간 경계 값들 "i_min" 및 "i_max"에 의해 정의된 검색 구간의 중심에 있는 테이블 "ari_hash_m[]"의 엔트리 "ari_hash_m[i]"가, 입력 변수 c에 의해 기술된 콘텍스트 값과 적어도 비슷하다. 그러므로, 입력 변수 c에 의해 기술된 콘텍스트 값이 테이블 "ari_hash_m[]"의 엔트리에 의해 기술된 유효 상태 값과 같지 않은 경우 외에는, 입력 변수 c에 의해 기술된 콘텍스트 값은 서브 알고리즘(506ba)의 반복 완료 이후에 "ari_hash_m[i_min]" 및 "ari_hash_m[i_max]"에 의해 정의된 구간 내에 입력 변수 c에 의해 기술된 콘텍스트 값이 있게 되는 것이 달성된다.

[0211] 만약, 그러나, ("i_max - i_min"에 의해 정의된) 구간의 크기가 그 최소 값에 도달하거나 초과하여 서브 알고리즘(506ba)의 반복적인 되풀이가 종료되면, 입력 변수 c에 의해 기술된 콘텍스트 값이 유효 상태 값이 아니라고 추정된다. 이 경우에, 구간의 상부 경계를 지칭하는 인덱스 "i_max"가, 그럼에도 불구하고, 이용된다. 서브 알고리즘(506ba)의 마지막 반복에서 도달되는 구간의 상부 값 "i_max"은 테이블 "ari_lookup_m"에 접근하기 위한 테이블 인덱스 값으로 재이용된다. 테이블 "ari_lookup_m[]"은 복수의 인접한 수치적 콘텍스트 값들의 구간들과 연관된 맵핑 규칙 인덱스 값들을 기술한다. 테이블 "ari_lookup_m[]"의 엔트리들에 의해 기술되는 맵핑 규칙 인덱스 값들이 연관되는 구간들은 테이블 "ari_lookup_m[]"의 엔트리들에 의해 기술된 유효 상태 값들에 의해 정의된다. 테이블 "ari_hash_m"의 엔트리들은 유효 상태 값들 및 인접한 수치적 콘텍스트 값의 구간들의 구간 경계들을 모두 정의한다. 알고리즘(506b)의 실행에서, 입력 변수 c에 의해 기술된 수치적 콘텍스트 값이 유효 상태 값과 같은지 여부, 및 만약 그 경우가 아니라면, (그 경계들이 유효 상태 값들에 의해 정의되는 복수의 구간들 중에서) 수치적 콘텍스트 값들의 어느 구간에 입력 변수 c에 의해 기술된 콘텍스트 값이 있는지가 결정된다. 그러므로, 알고리즘 506b는 입력 변수 c가 유효 상태 값을 기술하는지 여부를 결정하고, 만약 그 경우가 아니라면, 입력 변수 c에 의해 표현된 콘텍스트 값이 있는, 유효 상태 값들에 의해 경계지어진 구간을 식별하는 두 가지 기능을 만족시킨다. 이에 따라, 알고리즘 506e는 특히 효율적이고, 단지 비교적 적은 테이블 접근 횟수를 요구한다.

[0212] 상기를 요약하면, 콘텍스트 상태 c는 최상위 2 비트 방식(wise) 평면 m 의 디코딩을 위해 이용되는 누적 빈도 테이블을 결정한다. c로부터 상응하는 누적 빈도 테이블 인덱스 "pki"로의 맵핑이 함수 "arith_get_pk()"에 의해 수행된다. 함수 "arith_get_pk()"의 의사 프로그램 코드 표현이 도 5e를 참조하여 설명되었다.

[0213] 상기를 더 요약하면, 값 m이 누적 빈도 테이블 "arith_cf_m[pki][]"를 갖는 호출된 (하기에서 더욱 상세히 기술되는) 함수 "arith_decode()"를 이용하여 디코딩되는데, 여기서 "pki"는 도 5e를 참조하여 기술되는 함수 "arith_get_pk()"에 의해 반환된 (맵핑 규칙 인덱스 값이라고도 지칭되는) 인덱스에 상응한다.

[0214] 11.5.2 도 5f에 따른 알고리즘을 이용하는 맵핑 규칙 선택

[0215] 다음에서는, 스펙트럼 값들의 튜플의 디코딩에 이용될 수 있는 그러한 알고리즘에 대한 의사 프로그램 코드 표현을 도시하는 도 5f를 참조하여 맵핑 규칙 선택 알고리즘 "arith_get_pk()"에 대한 다른 실시예가 기술될 것이다. 도 5f에 따른 알고리즘은 알고리즘 "get_pk()", 또는 알고리즘 "arith_get_pk()"의 최적화된 버전(예를 들어, 속도 최적화 버전)으로 여겨질 수 있다.

[0216] 도 5f에 따른 알고리즘 "arith_get_pk()"은, 입력 변수로써, 콘텍스트의 상태를 기술하는 변수 c를 수신한다. 입력 변수 c는, 예를 들어, 수치적 현재 콘텍스트 값을 표현한다.

[0217] 알고리즘 "arith_get_pk()"은, 출력 변수로써, 입력 변수 c에 의해 기술된 콘텍스트의 상태에 연관된 확률 분포(또는 확률 모델)의 인덱스를 기술하는 변수 "pki"를 제공한다. 변수 "pki"는, 예를 들어, 맵핑 규칙 인덱스 값일 수 있다.

[0218] 도 5f에 따른 알고리즘은 어레이 "i_diff[]"의 콘텐츠에 대한 정의를 포함한다. 알 수 있는 바와 같이, (어레이 인덱스 0을 갖는) 어레이 "i_diff[]"의 제1 엔트리는 299와 동일하고, (어레이 인덱스들 1 내지 8을 갖는) 추가 어레인 엔트리들은 149, 74, 37, 18, 9, 4, 2, 및 1의 값들을 취한다. 이에 따라, 어레이들 "i_diff[]"의 엔트리들이 스텝 크기들을 정의하므로, 해시 테이블 인덱스 값 "i_min"의 선택을 위한 스텝 크기들이 각각의 반복에 따라 감소된다. 세부적인 사항들을 위해, 하기 논의가 참조된다.

[0219] 그러나, 각각 다른 스텝 크기들은, 예를 들어, 어레이 "i_diff[]"의 각각 다른 콘텐츠들은 실제로 선택될 수 있는데, 여기서 어레이 "i_diff[]"의 콘텐츠들은 해시 테이블 "ari_hash_m[i]"의 크기에 당연히 적응될 수 있다.

[0220] 변수 "i_min"는 알고리즘 "arith_get_pk()"의 시작에서 오른쪽에 0 값을 취하도록 초기화된다는 것을 알아야 한다.

[0221] 초기화 단계(508a)에서, 변수 s는 입력 변수 c에 따라 초기화되는데, 여기서 변수 c의 수치 표현은 변수 s의 수치 표현을 획득하기 위해 8 비트 만큼 왼쪽으로 이동된다.

[0222] 이어서, 해시 테이블 "ari_hash_m[]"의 엔트리의 해시 테이블 인덱스 값 "i_min"을 식별하기 위해 테이블 검색(508b)이 수행되어, 해시 테이블 엔트리 "ari_hash_m[i_min]"에 의해 기술된 콘텍스트 값과 다른 엔트리 "ari_hash_m"가 해시 테이블 엔트리 "ari_hash_m[i_min]"에 (그 해시 테이블 인덱스 값의 측면에서) 인접하는 다른 해시 테이블 엔트리 "ari_hash_m"에 의해 기술된 콘텍스트 값에 의해 경계지어지는 구간 내에 콘텍스트 값 c에 의해 기술된 콘텍스트 값이 있게 된다. 그러므로, 알고리즘 508b는 해시 테이블 "ari_hash_m[]"의 엔트리 "j=ari_hash_m[i_min"을 지칭하는 해시 테이블 인덱스 값 "i_min"의 결정을 가능하게 하여, 해시 테이블 엔트리 "ari_hash_m[i_min]"가 입력 변수 c에 의해 기술된 콘텍스트 값과 적어도 비슷하다.

[0223] 테이블 검색(508b)은 서브 알고리즘(508ba)의 반복적인 실행을 포함하는데, 여기서 서브 알고리즘(508ba)은 미리 결정된 횟수, 예를 들어, 9회 반복을 위해 실행된다. 서브 알고리즘(508ba)의 제1 단계에서, 변수 i는 변수 "i_min"의 값의 합과 같은 값 및 테이블 엔트리 "i_diff[k]"의 값으로 설정된다. 여기서 k는, 서브 알고리즘(508ba)의 각각의 반복에 따라, 초기 값 k=0으로부터 시작하여 증가되는 연속 변수임을 알아야 한다. 어레이 "i_diff[]"는 미리 결정된 증가 값들을 결정하는데, 여기서 증가 값들은 테이블 인덱스 k가 증가함에 따라, 즉, 반복 횟수가 증가함에 따라 감소한다.

[0224] 서브 알고리즘(508ba)의 제2 단계에서, 테이블 엔트리 "ari_hash_m[]"의 값은 변수 j에 복사된다. 바람직하게는, 테이블 "ari_hash_m[]"의 테이블 엔트리들의 최상위 비트들은 수치적 콘텍스트 값의 유효 상태 값들을 기술하고, 테이블 "ari_hash_m[]"의 엔트리들의 최하위 비트들(비트 0 내지 7)은 각각의 유효 상태 값들과 연관된 맵핑 규칙 인덱스 값들을 기술한다.

[0225] 서브 알고리즘(508ba)의 제3 단계에서, 변수 s의 값은 변수 j의 값과 비교되고, 만약 변수 s의 값이 변수 j의

값보다 더 크면 변수 "i_min"는 값 "i+1"로 선택적으로 설정된다. 이어서, 서브 알고리즘(508ba)의 제1 단계, 제2 단계, 및 제3 단계가 미리 결정된 횟수, 예를 들어, 9회 동안 반복된다. 그러므로, 서브 알고리즘(508ba)의 각각의 실행에서, 만약, 오직, 현재 유효한 해시 테이블 인덱스 "i_min + i_diff[]"에 의해 기술된 콘텍스트 값이 입력 변수 c에 의해 기술된 콘텍스트 값보다 더 작다면, 변수 "i_min"의 값이 i_diff[]+1 만큼 증가된다. 이에 따라, 만약(그리고 오직) 입력 변수 c, 및 결과적으로, 변수 s에 의해 기술된 콘텍스트 값이 엔트리 "ari_hash_m[i=i_min + diff[k]]"에 의해 기술된 콘텍스트 값보다 더 크면, 해시 테이블 인덱스 값 "i_min"이 서브 알고리즘(508ba)의 각각의 실행에서 (반복적으로) 증가된다.

[0226] 또한, 오직 단일 비교, 즉, 변수 s의 값이 변수 j의 값보다 더 큰지 여부에 관한 비교만이 서브 알고리즘(508ba)의 각각의 실행에서 수행됨을 알아야 한다. 이에 따라, 상기 알고리즘(508ba)은 계산에 관해 특히 효율적이다. 또한, 변수 "i_min"의 최종 값에 대하여 각각 다른 가능한 결과들이 있음을 알아야 한다. 예를 들어, 서브 알고리즘(508ba)의 마지막 실행 이후에 변수 "i_min"의 값이, 테이블 엔트리 "ari_hash_m[i_min]"에 의해 기술된 콘텍스트 값이 입력 변수 c에 의해 기술된 콘텍스트 값보다 더 작고, 테이블 엔트리 "ari_hash_m[i_min +1]"에 의해 기술된 콘텍스트 값이 입력 변수 c에 의해 기술된 콘텍스트 값보다 더 큰 것이 가능하다. 그렇지 않으면, 서브 알고리즘(508ba)의 마지막 실행 이후에, 해시 테이블 엔트리 "ari_hash_m[i_min -1]"에 의해 기술된 콘텍스트 값이 입력 변수 c에 의해 기술된 콘텍스트 값보다 더 작고, 엔트리 "ari_hash_m[i_min]"에 의해 기술된 콘텍스트 값이 입력 변수 c에 의해 기술된 콘텍스트 값보다 더 크게 될 수도 있다. 그렇지 않으면, 그러나, 해시 테이블 엔트리 "ari_hash_m[i_min]"에 의해 기술된 콘텍스트 값이 입력 변수 c에 의해 기술된 콘텍스트 값과 같게 될 수 있다.

[0227] 이러한 이유로, 결정에 기반한 반환 값 제공(508c)이 수행된다. 변수 j는 해시 테이블 엔트리 "ari_hash_m[i_min]"의 값을 취하도록 설정된다. 이어서, 입력 변수 c(및 또한 변수 s)에 의해 기술된 콘텍스트 값이 엔트리 "ari_hash_m[i_min]"에 의해 기술된 콘텍스트 값보다 더 큰지 여부(조건 "s>j"에 의해 정의된 제1 경우), 또는 입력 변수 c에 의해 기술된 콘텍스트 값이 해시 테이블 엔트리 "ari_hash_m[i_min]"에 의해 기술된 콘텍스트 값보다 더 작은지 여부(조건 "c<j>>8"에 의해 정의된 제2 경우), 또는 입력 변수 c에 의해 기술된 콘텍스트 값이 엔트리 "ari_hash_m[i_min]"에 의해 기술된 콘텍스트 값과 같은지 여부(제3 경우)가 결정된다.

[0228] 제1 경우(s>j)에서, 테이블 인덱스 값 "i_min+1"에 의해 지칭된 테이블 "ari_lookup_m[]"의 엔트리 "ari_lookup_m[i_min +1]"는 함수 "arith_get_pk()"의 출력 값으로써 반환된다. 제2 경우(c<(j>>8))에서, 테이블 인덱스 값 "i_min"에 의해 지칭된 테이블 "ari_lookup_m[]"의 엔트리 "ari_lookup_m[i_min]"는 함수 "arith_get_pk()"의 반환 값으로써 반환된다. 제3 경우(즉, 만약 입력 변수 c에 의해 기술된 콘텍스트 값이 테이블 엔트리 "ari_hash_m[i_min]"에 의해 기술된 유효 상태 값과 같으면), 해시 테이블 엔트리 "ari_hash_m[i_min]"의 최하위 8 비트에 의해 기술된 맵핑 규칙 인덱스 값은 함수 "arith_get_pk()"의 반환 값으로써 반환된다.

[0229] 상기를 요약하면, 508b 단계에서 특히 간단한 테이블 검색이 수행되는데, 여기서 테이블 검색은 입력 변수 c에 의해 기술된 콘텍스트 값이 테이블 "ari_hash_m[]"의 상태 엔트리들 중 하나에 의해 정의된 유효 상태 값과 같은지 아닌지 여부를 구분하지 않고 변수 "i_min"의 변수 값을 제공한다. 테이블 검색(508b) 단계에 이어 수행되는 508c 단계에서, 입력 변수 c에 의해 기술된 콘텍스트 값과 해시 테이블 엔트리 "ari_hash_m[i_min]"에 의해 기술된 유효 상태 값 사이의 크기 관계가 평가되고, 함수 "arith_get_pk()"의 반환 값이 상기 평가의 결과에 따라 선택되는데, 여기서 입력 변수 c에 의해 기술된 콘텍스트 값이 해시 테이블 엔트리 "ari_hash_m[i_min]"에 의해 기술된 유효 상태 값과 다르더라도, 테이블 평가(508b)에서 결정되는 변수 "i_min"의 값이 맵핑 규칙 인덱스 값을 선택하기 위해 고려된다.

[0230] 알고리즘에서의 비교는 바람직하게는 (또는, 그렇지 않으면) 콘텍스트 인덱스(수치적 콘텍스트 값) c와 j=ari_hash_m[i]>>8 사이에서 행해져야 함을 추가로 알아야 한다. 사실, 테이블 "ari_hash_m[]"의 각각의 엔트리는 8 번째 비트 이후에 코딩된 콘텍스트 인덱스, 및 맨 처음에서 8 비트(8 first bit, 최하위 비트들)로 코딩

된 그것의 상응하는 확률 모델을 표현한다. 현재의 구현에서, 우리는 주로, s=c<<8이 또한 ari_hash_m[i]보다 더 큰지를 감지하는 것과 같은, 현재 콘텍스트 c가 ari_hash_m[i]>>8보다 더 큰지 여부를 아는 것에 관심이 있다.

[0231] 상기를 요약하면, 일단 콘텍스트 상태가 계산되면(이는, 예를 들어, 도 5c에 따른 알고리즘 "arith_get_context(c,i,N)", 또는 도 5d에 따른 알고리즘 "arith_get_context(c,i)"을 이용하여 달성될 수 있다), 콘텍스트 상태에 상응하는 확률 모델에 상응하는 적절한 누적 빈도 테이블을 갖는 호출된 (하기에서 기술될) 알고리즘 "arith_decode"을 이용하여 최상위 2 비트 방식 평면이 디코딩된다. 함수 "arith_get_pk()", 예를 들어, 도 5f를 참조하여 논의된 함수 "arith_get_pk()"에 의해 관련성이 생긴다.

[0232] 11.6 산술 디코딩

[0233] 11.6.1 도 5g에 따른 알고리즘을 이용하는 산술 디코딩

[0234] 다음에서, 함수 "arith_decode()"의 기능이 도 5g를 참조하여 상세히 논의될 것이다.

[0235] 함수 "arith_decode()"는, 시퀀스의 제1 심볼이면 TRUE를, 그렇지 않으면 FALSE를 반환하는 조력 함수(helper fuction) "arith_first_symbol (void)"를 이용한다는 것을 알아야 한다. 함수 "arith_decode()"는 또한 비트스트림의 다음 비트를 받아서 제공하는 조력 함수 "arith_get_next_bit(void)"를 이용한다.

[0236] 또한, 함수 "arith_decode()"는 전역 변수들 "낮음", "높음", 및 "값"을 이용한다. 나아가, 함수 "arith_decode()"는, 입력 변수로써, 선택된 누적 빈도 테이블 또는 누적 빈도 테이블 서브 테이블의 (성분 인덱스 또는 엔트리 인덱스 0을 갖는) 제1 엔트리 또는 성분을 가리키는 변수 "cum_freq[]"을 수신한다. 또한, 함수 "arith_decode()"는 변수 "cum_freq[]"에 의해 지칭된 선택된 누적 빈도 테이블 또는 누적 빈도 서브 테이블의 길이를 가리키는 입력 변수 "cfl"를 이용한다.

[0237] 함수 "arith_decode()"는, 제1 단계로써, 만약 조력 함수 "arith_first_symbol()"가 심볼들의 스퀀스의 제1 심볼이 디코딩된다고 가리키면 수행되는 변수 초기화(570a)를 포함한다. 값 초기화(550a)는, 복수의, 예를 들어, 조력 함수 "arith_first_symbol()"를 이용하여 비트스트림으로부터 획득되는 16비트에 따라 변수 "value"을 초가화하여, 변수 "value"이 상기 비트들에 의해 표현된 값을 취한다. 또한, 변수 "low"는 값 0을 취하도록 초기화되고, 변수 "high"는 값 65535을 취하도록 초기화된다.

[0238] 제2 단계(570b)에서, 변수 "range"는 변수들 "high"과 "low"의 값들 사이의 차이보다 1 만큼 더 큰 값으로 설정된다. 변수 "cum"은 변수 "low"의 값과 변수 "high"의 값 사이의 변수 "value"의 값의 상대적 위치를 표현하는 값으로 설정된다. 이에 따라, 변수 "cum"은, 예를 들어, 변수 "value"의 값에 따라 0과 2^{16} 사이의 값을 취한다.

[0239] 포인터 p는 선택된 누적 빈도 테이블의 시작 주소보다 1 만큼 더 작은 값으로 초기화된다.

[0240] 알고리즘 "arith_decode()"은 또한 반복적인 누적 빈도 테이블 검색(570c)을 포함한다. 반복적인 누적 빈도 테이블 검색은 변수 cfl이 1보다 더 작거나 1과 같아질 때까지 반복된다. 반복적인 누적 빈도 테이블 검색(570c)에서, 포인터 변수 q는 포인터 변수 p의 현재 값과 변수 "cfl"의 값의 반의 합과 같은 값으로 설정된다. 만약

엔트리가 포인터 변수 q에 의해 어드레스되는 선택된 누적 빈도 테이블의 엔트리 *q의 값이 변수 "cum"의 값보다 더 크다면, 포인터 변수 p는 포인터 변수 q의 값으로 설정되고, 변수 "cfl"은 증가된다. 마지막으로, 변수 "cfl"이 1 비트 만큼 오른쪽으로 이동되어, 변수 "cfl"의 값을 2로 효과적으로 나누고 모듈로(modulo) 부분은 무시한다.

[0241] 이에 따라, 반복적인 누적 빈도 테이블 검색(570c)은, 누적 빈도 테이블의 엔트리들에 의해 경계지어지는 선택된 누적 빈도 테이블 내의 구간을 식별하기 위해, 변수 "cum"의 값을 선택된 누적 빈도 테이블의 복수의 엔트리들과 효과적으로 비교하여, 값 cum이 식별된 구간 내에 있게 된다. 이에 따라, 선택된 누적 빈도 테이블의 엔트리들은 구간들을 정의하는데, 여기서 각각의 심볼 값은 선택된 누적 빈도 테이블의 각각의 구간들에 연관된다. 또한, 누적 빈도 테이블의 두 개의 인접한 값들 사이의 구간들의 폭들을 상기 구간들과 연관된 심볼들의 확률을 정의하여, 선택된 누적 빈도 테이블 전체는 각각 다른 심볼들(또는 심볼 값들)의 확률 분포를 정의한다. 이용가능한 누적 빈도 테이블들에 관한 세부사항들이 도 23을 참조하여 하기에서 논의될 것이다.

[0242] 다시 도 5g를 참조하면, 심볼 값은 포인터 변수 p의 값으로부터 도출되는데, 여기서 심볼 값은 도면 부호 570d에서 도시된 바와 같이 도출된다. 그러므로, 변수 "symbol"에 의해 표현되는 심볼 값을 획득하기 위해, 포인터 변수 p와 시작 주소 "cum_freq"의 값 사이의 차이가 평가된다.

[0243] 알고리즘 "arith_decode"은 또한 변수들 "high" 및 "low"의 적응(adaptation, 570e)을 포함한다. 만약 변수 "symbol"에 의해 표현된 심볼 값이 0과 다르다면, 도면 부호 570e에 도시된 바와 같이, 변수 "high"가 업데이트된다. 또한, 도면 부호 570e에 도시된 바와 같이, 변수 "low"의 값이 업데이트된다. 변수 "high"는 변수 "low", 변수 "range", 및 변수 "range", 및 선택된 누적 빈도 테이블의 인덱스 "symbol-1"을 갖는 엔트리에 의해 결정되는 값으로 설정된다. 변수 "low"는 증가되는데, 여기서 증가 크기는 변수 "range" 및 인덱스 "symbol"를 갖는 선택된 누적 빈도 테이블의 엔트리에 의해 결정된다. 이에 따라, 변수들 "low"와 "high"의 값들 사이의 차이는 선택된 누적 빈도 테이블의 두 개의 인접한 엔트리들 사이의 수치 차이에 따라 조절된다.

[0244] 이에 따라, 만약 낮은 확률을 갖는 심볼 값이 감지된다면, 변수들 "low" 및 "high"의 값들 사이의 구간이 좁은 폭으로 감소된다. 그에 반해서, 만약 감지된 심볼 값이 상대적으로 큰 확률을 포함한다면, 변수들 "low" 및 "high"의 값들 사이의 구간의 폭이 비교적 큰 값으로 설정된다. 다시, 변수들 "low" 및 "high"의 값들 사이의 구간의 폭은 감지된 심볼 및 누적 빈도 테이블의 상응하는 엔트리들에 따른다.

[0245] 알고리즘 "arith_decode()"은 또한, 570e 단계에서 결정된 구간이 "break" 조건에 도달될 때까지 반복적으로 이동되고 스케일링되는 구간 재정상화(renormalization, 570f)를 포함한다. 구간 재정상화(570f)에서, 선택적 하향 이동 연산(570fa)이 수행된다. 만약 변수 "high"가 32768보다 더 작다면, 아무것도 행해지지 않고, 구간 재정상화는 구간 크기 증가 연산(570fb)을 계속한다. 만약, 그러나, 변수 "high"가 32768보다 더 작지 않고 변수 "low"가 32768보다 더 크거나 32768과 같다면, 변수들 "values", "low", 및 "high"는 모두 32768로 감소되어, 변수들 "low" 및 "high"에 의해 정의된 구간이 하향 이동되고, 변수 "value"의 값이 또한 하향 이동된다. 만약, 그러나, 변수 "high"의 값이 32768보다 더 작지 않고, 변수 "low"가 32768보다 더 크지 않거나 32768과 같고, 변수 "low"가 16384보다 더 크거나 16384와 같고, 변수 "high"가 49152보다 더 작은 것으로 확인되면, 변수들 "value", "low", 및 "high"는 모두 16384로 감소되어, 변수들 "high"와 "low"의 값들 사이의 구간 및 또한 변수 "value"의 값이 하향 이동된다. 만약, 그러나, 상기 조건들 중 어느 것도 충족되지 않는다면, 구간 재정상화는 중단된다.

[0246] 만약, 그러나, 570fa 단계에서 평가되는, 상기에서 언급된 조건들 중 어느 것이 만족된다면, 구간 증가 연산(570fb)이 실행된다. 구간 증가 연산(570fb)에서, 변수 "low"의 값은 두 배가 된다. 또한, 변수 "high"의 값이 두 배가 되고, 두 배가 된 결과 1 만큼 증가된다. 또한, 변수 "value"의 값이 두 배가 되고(1 비트 만큼 왼쪽으로 이동되고), 조력 함수 "arith_get_next_bit"에 의해 획득되는 비트스트림의 한 비트는 최하위 비트로 이용된

다. 이에 따라, 변수들 "low" 및 "high"의 값들 사이의 구간의 크기는 대략 두 배로 되고, 변수 "value"의 정확도는 비트스트림의 새로운 비트를 이용하여 증가된다. 상기에서 언급한 바와 같이, 570fa 및 570fb 단계들은 "break" 조건에 도달될 때까지, 즉, 변수들 "low" 및 "high"의 값들 사이의 구간이 충분히 클 때까지 반복된다.

[0247] 알고리즘 "arith_decode()"의 기능과 관련하여, 변수들 "low" 및 "high"의 값들 사이의 구간은 변수 "cum_freq"에 의해 참조된 누적 빈도 테이블의 두 개의 인접한 엔트리들에 따라 570e 단계에서 감소된다는 것을 알아야 한다. 만약 선택된 누적 빈도 테이블의 두 개의 인접한 값들 사의 구간이 작다면, 즉, 만약 인접한 값들이 비교적 서로 가깝다면, 570e 단계에서 획득되는 변수들 "low"와 "high"의 값들 사이의 구간이 비교적 작을 것이다. 그에 반해서, 누적 빈도 테이블의 두 개의 인접한 엔트리들이 더 멀게 구간이 띄어진다면, 570e 단계에서 획득되는 변수들 "low"와 "high"의 값들 사이의 구간이 비교적 클 것이다.

[0248] 결과적으로, 570e 단계에서 획득되는 변수들 "low"와 "high"의 값들 사이의 구간이 비교적 작다면, (조건 평가(570fa)의 어떠한 조건들도 충족되지 않도록) "충분한" 크기로 구간을 재스케일링하기 위해 많은 횟수의 재정상화 단계들이 실행될 것이다. 이에 따라, 변수 "value"의 정확도를 증가시키기 위해 비스트스트림으로부터 비교적 많은 수의 비트들이 이용될 것이다. 만약, 반대로, 570e 단계에서 획득된 구간 크기가 비교적 크다면, 변수들 "low" 및 "high"의 값들 사이의 구간을 "충분한" 크기로 재정상화하기 위해 더 적은 횟수의 구간 재정상화 단계들(570fa 및 570fb)의 반복만이 요구될 것이다. 이에 따라, 변수 "value"의 정확도를 증가시키고 다음 심볼의 디코딩을 준비하기 위해 비트스트림으로부터 오직 비교적 적은 수의 비트들만이 이용될 것이다.

[0249] 상기를 요약하면, 만약 비교적 높은 가능성을 포함하고, 큰 구간이 선택된 누적 빈도 테이블의 엔트리들에 연관되는 심볼이 디코딩되면, 이어지는 심볼들의 디코딩을 가능하게 하기 위해 오직 비교적 적은 수의 비트들만이 비트스트림으로부터 판독될 것이다. 그에 반해서, 만약 비교적 적은 확률을 포함하고 작은 구간이 선택된 누적 빈도 테이블의 엔트리들에 의해 연관되는 심볼이 디코딩된다면, 다음 심볼의 디코딩을 준비하기 위해 비교적 많은 수의 비트들이 비트스트림으로부터 취해질 것이다.

[0250] 이에 따라, 누적 빈도 테이블의 엔트리들은 각각 다른 심볼들의 확률을 반영하고, 또한 심볼들의 시퀀스를 디코딩하기 위해 요구되는 비트들의 수를 반영한다. 콘텍스트에 따라, 즉, 이전에 디코딩된 심볼들(또는 스펙트럼 값들)에 따라 누적 빈도 테이블을 다르게 하여, 예를 들어, 콘텍스트에 따라 각각 다른 누적 빈도 테이블들을 선택하여, 각각 다른 심볼들 사이의 확률적인 의존성이 이용될 수 있는데, 이는 이어지는 (또는 인접한) 심볼들에 대한 특정한 비트율 효율적인 인코딩을 가능하게 한다.

[0251] 상기를 요약하면, (반환 변수 "symbol"에 의해 표현된 심볼 값으로 설정될 수 있는) 최상위 비트 평면 값 m을 결정하기 위해 함수 "arith_get_pk()"에 의해 반환된 인덱스 "pki"에 상응하는 누적 빈도 테이블 "arith_cf_m[pki][]"과 함께 도 5g를 참조하여 기술된 함수 "arith_decode()"가 호출된다.

[0252] 상기를 요약하면, 산술 디코더는 스케일링과 함께 태그 생성 방법을 이용하는 정수 구현이다. 세부적인 사항들을 위해, K. Sayood의 "데이터 압축 입문서(Introduction to Data Compression)", 제3판, 2006, Elsevier Inc.라는 책이 참조된다.

[0253] 도 5g에 따른 컴퓨터 프로그램 코드는 본 발명의 일 실시예에 따라 이용된 알고리즘을 기술한다.

[0254] 11.6.2 도 5h 및 5i에 따른 알고리즘을 이용하는 산술 디코딩

[0255] 도 5h 및 5i는 알고리즘 "arith_decode()"의 다른 실시예에 대한 의사 프로그램 코드 표현을 도시하는데, 이는 도 5g를 참조하여 기술된 알고리즘 "arith_decode"의 대안으로 이용될 수 있다.

[0256] 도 5g 및 도 5h와 5i에 따른 알고리즘들은 모두 도 3에 따른 알고리즘 "values_decode()"에서 이용될 수 있음을 알아야 한다.

[0257] 요약하면, 값 m은 누적 빈도 테이블 "arith_cf_m[pki][]"을 갖는 호출된 함수 "arith_decode()"를 이용하여 디코딩되는데, 여기서 "pki"는 함수 "arith_get_pk()"에 의해 반환된 인덱스에 상응한다. 산술 코더(또는 디코더는) 스케일링과 함께 태그 생성 방법을 이용하는 정수 구현이다. 세부적인 사항들을 위해, K. Sayood의 "데이터 압축 입문서", 제3판, 2006, Elsevier Inc.라는 책이 참조된다. 도 5h 및 5i에 따른 컴퓨터 프로그램은 이용된 알고리즘을 기술한다.

[0258] 11.7 이스케이프 매커니즘

[0259] 다음에서는, 도 3에 따른 디코딩 알고리즘 "values_decode()"에서 이용되는 이스케이프 매커니즘이 간략히 논의될 것이다.

[0260] (함수 "arith_decode()"의 반환 값으로써 제공되는) 디코딩된 값 m이 이스케이프 심볼 "ARITH_ESCAPE"일 때, 변수들 "lev" 및 "esc_nb"은 1 만큼 증가되고, 다른 값 m은 디코딩된다. 이 경우에, 입력 인수로써 값 "c+esc_nb<<17"을 갖는 함수 "arith_get_pk()"이 다시 한번 호출되는데, 변수 "esc_nb"는 동일한 2-튜플에 대해 이전에 디코딩되고 7 개로 경계지어진 이스케이프 심볼들의 수를 기술한다.

[0261] 요약하면, 만약 이스케이프 심볼이 식별되면, 최상위 비트 평면 값 m은 증가된 수치적 가중치를 포함하는 것으로 여겨진다. 또한, 현재 수치적 디코딩이 반복되는데, 여기서 수정된 수치적 현재 콘텍스트 값 "c+esc_nb<<17"은 함수 "arith_get_pk()"에 입력 변수로써 이용된다. 이에 따라, 각각 다른 맵핑 규칙 인덱스 값 "pki"는 일반적으로 서브 알고리즘(312ba)의 각각 다른 반복에서 획득된다.

[0262] 11.8 산술 중지 매커니즘

[0263] 다음에서는, 산술 중지 매커니즘이 기술될 것이다. 산술 중지 매커니즘은 오디오 인코더에서 상위 주파수 부분이 전체적으로 0으로 양자화되는 경우에 요구되는 비트들의 수의 감소를 가능하게 한다.

[0264] 일 실시예에서, 산술 중지 매커니즘은 다음과 같이 구현될 수 있다: 일단 값 m이 이스케이프 심볼 "ARITH_ESCAPE"이 아니면, 디코더는 연이은 m이 "ARITH_ESCAPE" 심볼인지를 검사한다. 만약 조건 "esc_nb>0&&m==0"이 참이라면, "ARITH_STOP" 심볼이 감지되고 디코딩 과정이 종료된다. 이 경우에, 디코더는 하기에서 기술될 "arith_finish()" 함수로 바로 점프한다. 상기 조건은 프레임의 나머지가 0 값들로 구성되는 것을 의미한다.

[0265] 11.9 하위 비트 평면 디코딩

[0266] 다음에서는, 하나 이상의 하위 비트 평면들에 대한 디코딩이 기술될 것이다. 하위 비트 평면의 디코딩은, 예를

들어, 도 3에 도시된 312d 단계에서 수행된다. 그렇지 않으면, 그러나, 도 5j 및 5n에 도시된 알고리즘들이 이용될 수 있다.

[0267] 11.9.1 도 5j에 따른 하위 비트 평면 디코딩

[0268] 이제 도 5j를 참조하면, 변수들 a 및 b의 값들이 변수 m으로부터 도출됨을 알 수 있다. 예를 들어, 변수 b의 수치 표현을 획득하기 위해 값 m의 수치 표현은 2 비트 만큼 오른쪽으로 이동된다. 또한, 변수 m의 값으로부터, 2 비트 만큼 왼쪽으로 이동된 변수 b의 값의 비트 이동된 버전을 빼서 변수 a의 값이 획득된다.

[0269] 이어서, 최하위 비트 평면 값들 r에 대한 산술 디코딩이 반복되는데, 여기서 반복 횟수는 변수 "lev"의 값에 의해 결정된다. 최하위 비트 평면 값 r은 함수 "arith_decod"를 이용하여 획득되는데, 여기서 최하위 비트 평면 디코딩에 적응된 누적 빈도 테이블이 이용된다(누적 빈도 테이블 "arith_cf_r"). 변수 r의 (수치적 가중치 1을 갖는) 최하위 비트는 변수 a에 의해 표현된 스펙트럼 값의 하위 비트 평면을 기술하고, 변수 r의 수치적 가중치 2를 갖는 비트는 변수 b에 의해 표현된 스펙트럼 값의 하위 비트를 기술한다. 이에 따라, 변수 a는 1 비트 만큼 왼쪽으로 변수 a를 이동시키고, 최하위 비트로써 변수 r의 수치적 가중치 1을 갖는 비트를 추가함으로써 업데이트된다. 유사하게, 변수 b는 1 비트 만큼 왼쪽으로 변수 b를 이동시키고, 변수 r의 수치적 가중치 2를 갖는 비트를 추가함으로써 업데이트된다.

[0270] 이에 따라, 변수들 a, b의 비트들을 전달하는 2 개의 최상위 정보가 최상위 비트 평면 값 m에 의해 결정되고, 값들 a 및 b의 (만약에 있다면) 하나 이상의 최하위 비트들은 하나 이상의 하위 비트 평면 값 r에 의해 결정된다.

[0271] 상기를 요약하면, 만약 "ARITH_STOP" 심볼을 충족시키지 않는다면, 잔여 비트 평면들이, 그 다음에, 디코딩되고, 만약 있다면, 현재 2-튜플이 디코딩된다. 잔여 비트 평면들은 누적 빈도 테이블 "arith_cf_r[]"을 갖는 함수 "arith_decode()"를 lev 횟수 호출하여 최상위로부터 최하위 레벨로 디코딩된다. 디코딩된 비트 평면들 r은 그 의사 프로그램 코드가 도 5j에 도시되는 알고리즘에 따라 이전에 디코딩된 값 m의 정제를 허용한다.

[0272] 11.9.2 도 5n에 따른 하위 비트 대역 디코딩

[0273] 그렇지 않으면, 그러나, 그 의사 프로그램 코드가 도 5n에 도시되는 알고리즘이 또한 하위 비트 평면 디코딩에 이용될 수 있다. 이 경우에, "ARITH_STOP"을 충족시키지 않는다면, 잔여 비트 평면들이, 그러면, 디코딩되고, 만약 있다면,현재 2-튜플이 디코딩된다. 잔여 비트 평면들은 누적 빈도 테이블 "arith_cf_r()"를 갖는 "arith_decode()"를 "lev" 회 호출하여 최상위로부터 최하위 레벨로 디코딩된다. 디코딩된 비트 평면들 r은 도 5n에 도시된 알고리즘에 따라 이전에 디코딩된 값 m의 정제를 허용한다.

[0274] 11.10 콘텍스트 업데이트

[0275] 11.10.1 도 5k, 5l, 및 5m에 따른 콘텍스트 업데이트

[0276] 다음에서는, 스펙트럼 값들의 튜플에 대한 디코딩을 완료하기 위해 이용된 연산들이 도 5k 및 5l을 참조하여 기술될 것이다. 또한, 오디오 콘텐츠의 현재 부분(예를 들어, 현재 프레임)과 연관된 스펙트럼 값들의 튜플들의 셋트에 대한 디코딩을 완료하기 위해 이용되는 연산이 기술될 것이다.

[0277] 이제 도 5k를 참조하면, 하위 비트 디코딩(312d) 이후에, 어레이 "x_ac_dec[]"의 엔트리 인덱스 2*i를 갖는 엔트리는 a와 같게 설정되고, 어레이 "x_ac_dec[]"의 엔트리 인덱스 "2*i+1"을 갖는 엔트리는 b와 같게 설정됨을 알 수 있다. 다시 말해서, 하위 비트 디코딩(312d) 이후의 지점에서, 2-튜플(a,b)의 무보호 값이 완전히 디코딩된다. 그것은 도 5k에 도시된 알고리즘에 따라 스펙트럼 계수들을 가지고 있는 성분(예를 들어, 어레이 "x_ac_dec[]") 안에 저장된다.

[0278] 이어서, 다음의 2-튜플들을 위해 콘텍스트 "q"가 또한 업데이트된다. 이러한 콘텍스트 업데이트는 마지막 2-튜플을 위해서도 또한 수행되어야 함을 알아야 한다. 이 콘텍스트 업데이트는 그 의사 프로그램 코드 표현이 도 5l에 도시되는 함수 "arith_update_context()"에 의해 수행된다.

[0279] 이제 도 5l을 참조하면, 함수 "arith_update_context(i,a,b)"는, 입력 변수들로써, 2-튜플의 디코딩된 무부호 양자화된 스펙트럼 계수들(또는 스펙트럼 값들) a, b를 수신함을 알 수 있다. 또한, 함수 "arith_update_contex"은 또한, 입력 변수로써, 디코딩하기 위한 양자화된 스펙트럼 계수의 인덱스 i(예를 들어, 주파수 인덱스)를 수신한다. 다시 말해서, 입력 변수 i는, 예를 들어, 그 절대 값들이 입력 변수들 a, b에 의해 정의되는 스펙트럼 값들의 튜플의 인덱스일 수 있다. 알 수 있는 바와 같이, 어레이 "q[][]"의 엔트리 "q[1][i]"는 a+b+1와 같은 값으로 설정될 수 있다. 또한, 어레이 "q[][]"의 엔트리 "q[1][i]"의 값은 16진 값 "0xF"으로 제한될 수 있다. 그러므로, 어레이 "q[][]"의 엔트리 "q[1][i]"는 주파수 인덱스 i를 갖는 스펙트럼 값들의 현재 디코딩된 튜플 {a,b}의 절대 값들의 합을 계산하고, 상기 합의 결과에 1을 추가함으로써 획득된다.

[0280] 여기서 어레이 "q[][]"의 엔트리 "q[1][i]"는 콘텍스트 서브구역 값으로 간주될 수 있음을 알아야 하는데, 이는 그것이 추가적인 스펙트럼 값들(또는 스펙트럼 값들의 튜플들)의 이어지는 디코딩에 이용되는 콘텍스트의 서브구역을 기술하기 때문이다.

[0281] 여기서, (그 부호 버전들이 어레이 "x_ac_dec[]"의 엔트리들 "x_ac_dec[2*i]" 및 "x_ac_dec[2*i+1]"에 저장되는) 2 개의 현재 디코딩된 스펙트럼 값들의 절대 값들 a 및 b의 합계는, 디코딩된 스펙트럼 값들의 놈(예를 들어, L1 놈)의 계산으로 여겨질 수 있음을 알아야 한다.

[0282] 복수의 이전에 디코딩된 스펙트럼 값들에 의해 형성된 벡터의 놈을 기술하는 콘텍스트 서브구역 값들(즉, 어레이 "q[][]"의 엔트리들)은 특히 의미 있고 메모리 효율적인 것으로 확인됐다. 복수의 이전에 디코딩된 스펙트럼 값들에 기초하여 계산되는 그러한 놈은 압축된(compact) 형태로 의미 있는 콘텍스트 정보를 포함하는 것으로 확인됐다. 스펙트럼 값들의 부호는 일반적으로 콘텍스트의 선택에 특별히 관련이 있지 않은 것으로 확인됐다. 또한, 복수의 이전에 디코딩된 스펙트럼 값들에 걸친 놈의 형성은 일반적으로, 몇몇 세부사항들이 버려질지라도, 가장 중요한 정보를 유지하는 것으로 확인됐다. 또한, 최대 값으로 수치적 현재 상태 값을 제한하는 것은 일반적으로 정보의 심각한 손실을 야기하지 않는 것으로 확인됐다. 오히려, 미리 결정된 임계 값보다 더 큰 유효 스펙트럼 값들에 대해 동일한 콘텍스트 상태를 사용하는 것이 더 효율적인 것으로 확인됐다. 그러므로, 콘텍스트 서브구역 값들의 제한은 메모리 효율의 추가적인 개선을 가져온다. 뿐만 아니라, 특정 최대 값들로 콘텍스트 서브구역 값들을 제한하는 것은 수치적 현재 콘텍스트 값의 특히 간단하고 계산 효율적인 업데이트를 가능하게 하는 것으로 확인됐는데, 이는, 예를 들어, 도 5c 및 5d를 참조하여 기술되었다. 비교적 작은 값(예를 들어, 값 15)으로 콘텍스트 서브구역 값들을 제한함으로써, 복수의 콘텍스트 서브구역 값들에 기초하는 콘텍스트 상태가 효율적인 형태로 표현될 수 있는데, 이는 도 5c 및 5d를 참조하여 논의되었다.

[0283] 또한, 1과 15 사이의 값들로 콘텍스트 서브구역 값들을 제한하는 것은 정확도와 메모리 효율성 사이에 특히 좋은 절충을 불러오는 것으로 확인됐는데, 이는 그러한 콘텍스트 서브구역 값을 저장하기 위해 4비트가 충분하기 때문이다.

[0284] 그러나, 몇몇 다른 실시예들에서, 콘텍스트 서브구역 값들은 오직 단일 디코딩된 스펙트럼 값에만 기초할 수 있음을 알아야 한다. 이 경우에, 놈의 형성이 선택적으로 생략될 수 있다.

[0285] 프레임의 다음 2-tuple은, 함수 "arith_get_context()"로부터 시작하여, 1 만큼 i를 증가시키고 상기에서 기술된 것과 동일한 과정을 다시 함으로써 함수 "arith_update_context"을 완료한 이후에 디코딩된다.

[0286] lg/2 2-튜플들이 프레임 내에서 디코딩되거나, "ARITH_ESCAPE"에 따른 중지 심볼이 발생할 때, 스펙트럼 진폭의 디코딩 과정이 종료되고 부호들의 디코딩이 시작된다.

[0287] 부호들의 디코딩에 관한 세부사항들이 도 3을 참조하여 논의되었는데, 여기서 부호들의 디코딩은 도면 부호 314에서 도시된다.

[0288] 일단 모든 무부호 양자화된 스펙트럼 계수들이 디코딩되면, 그에 따른 부호가 추가된다. 각각의 널이 아닌(non-null) 양자화된 값 "x_ac_dec"에 대해 하나의 비트가 판독된다. 만약 판독된 비트 값이 0과 같다면, 양자화된 값은 양(positive)이며, 아무것도 행해지지 않고, 부호 값은 이전에 디코딩된 무부호 값과 같다. 그렇지 않으면 (즉, 판독된 비트 값이 1과 같으면), 디코딩된 계수(또는 스펙트럼 값)은 음(negative)이고, 무보호 값으로부터 2의 보수가 취해진다. 부호 비트는 낮은 주파수들에서 높은 주파수까지 판독된다. 세부적인 사항들을 위해, 도 3 및 부호 디코딩(314)에 관한 설명이 참조된다.

[0289] 디코딩은 함수 "arith_finish()"를 호출함으로써 종료된다. 잔여 스펙트럼 계수들은 0으로 설정된다. 각각의 콘텍스트 상태들은 그에 상응하여 업데이트된다.

[0290] 세부적인 사항들을 위해, 함수 "arith_finish()"에 대한 의사 프로그램 코드 표현을 도시하는 도m이 참조된다. 알 수 있는 바와 같이, 함수 "arith_finish()"는 디코딩된 양자화 스펙트럼 계수들을 기술하는 입력 변수 lg를 수신한다. 바람직하게는, 함수 "arith_finish()"의 입력 변수 lg는, "ARITH_STOP" 심볼의 감지에 응답하여 0 값이 할당되는, 고려되지 않는 스펙트럼 계수들을 남겨 두며, 실제 디코딩된 스펙트럼 계수들의 수를 기술한다. 함수 "arith_finish()"의 입력 변수 N은 현재 윈도우(즉, 오디오 콘텐츠의 현재 부분과 연관된 윈도우)의 윈도우 길이를 기술한다. 일반적으로, 길이 N인 윈도우와 연관된 스펙트럼 값들의 수는 N/2와 같고, 윈도우 길이 N인 윈도우와 연관된 스펙트럼 값들의 2-튜플들의 수는 N/4와 같다.

[0291] 함수 "arith_finish"는 또한, 입력 값으로써, 디코딩된 스펙트럼 값들의 벡터 "x_ac_dec", 또는 적어도 디코딩된 스펙트럼 계수들의 그러한 벡터에 대한 참조를 수신한다.

[0292] 함수 "arith_finish"는 산술 중지 조건의 존재로 인해 어떠한 스펙트럼 값들도 디코딩되지 않는 어레이(또는 벡터) "x_ac_dec"의 엔트리들을 0으로 설정하도록 구성된다. 또한, 함수 "arith_finish"는, 산술 중지 조건의 존재로 인해 어떠한 값도 디코딩되지 않는 스펙트럼 값들에 연관되는 콘텍스트 서브구역 값들 "q[1][i]"을 미리 결정된 값 1로 설정한다. 미리 결정된 값 1은 스펙트럼 값들의 튜플에 상응하는데, 여기서 스펙트럼 값들은 모두 0과 같다.

[0293] 이에 따라, 산술 중지 조건이 존재하더라도, 함수 "arith_finish()"은 스펙트럼 값들의 전체 어레이(또는 백터) "x_ac_dec[]", 및 또한 콘텍스트 서부 구역 값들"q[1][i]"의 전체 어레이를 업데이트하는 것을 가능하게 한다.

[0294] 11.10.2 도 5o 및 5p에 따른 콘텍스트 업데이트

[0295] 다음에서는, 도 5o 및 5p를 참조하여 콘텍스트 업데이트에 대한 다른 실시예가 기술될 것이다. 2-튜플 (a,b)의 무부호 값이 완전히 디코딩되는 지점에서, 콘텍스트 q가, 그 다음에, 다음 2-튜플을 위해 업데이트된다. 상기 업데이트는 만약 현재 2-튜플이 마지막 2-튜플이라도 수행된다. 업데이트들은 모두 그 의사 프로그램 코드 표현이 도 5o에 도시되는 함수 "arith_update_context()"에 의해 이루어진다.

[0296] 프레임의 다음 2-튜플은, 그 다음에, 1 만큼 i를 증가시키고 함수 arith_decode()을 호출함으로써 디코딩된다. 만약 lg/2 2-튜플들이 이미 프레임으로 디코딩되었거나, 만약 중지 심볼 "ARITH_STOP"이 발생했다면, 함수 "arith_finish()"이 호출된다. 상기 콘텍스트는 다음 프레임을 위해 어레이 (또는 벡터) "qs"에 보관되어 저장된다. 함수 "arith_save_context()"의 의사 프로그램 코드가 도 5p에 도시된다.

[0297] 일단 모두 무부호 양자화된 스펙트럼 계수들이 디코딩되면, 부호가 그 다음에 추가된다. 각각의 양자화되지 않은 값 "qdec"에 대해, 하나의 비트가 판독된다. 만약 판독된 비트가 0과 같으면, 양자화된 값은 양이며, 아무것도 행해지지 않고, 부호 값은 이전에 디코딩된 무부호 값과 같다. 그렇지 않으면, 디코딩된 계수는 음이고, 무보호 값으로부터 2의 보수가 취해진다. 부호 비트들은 낮은 주파수들에서 높은 주파수들까지 판독된다.

[0298] 11.11 디코딩 과정에 대한 요약

[0299] 다음에서는, 디코딩 과정이 간략히 요약될 것이다. 세부적인 사항들을 위해, 상기 논의와 또한 도 3, 4, 5a, 5c, 5e, 5g, 5j, 5k, 5l, 및 5m이 참조된다. 양자화된 스펙트럼 계수들 "x_ac_dec[]"은 가장 낮은 주파수 계수에서 시작하여 가장 높은 주파수 계수까지 나아가며 무잡음 디코딩된다. 그것들은 이른바 2-튜플 (a,b)로 모이는 2 개의 연이은 계수들 a,b의 그룹들에 의해 디코딩된다.

[0300] 주파수 도메인(즉, 주파수 도메인 모드)에 대해 디코딩된 계수들 "x_ac_dec[]"은, 그 다음에, 어레이 "x_ac_quant[g][win][sfb][bin]"에 저장된다. 무잡음 코딩 코드워드들의 전송 순서는, 그것들이 어레이에 수신되어 저장된 순서로 디코딩될 때, "bin"이 가장 빠르게 증가하는 인덱스이고 "g"가 가장 느리게 증가하는 인덱스이다. 코드워드 내에서, 디코딩 순서는, a, 그 다음에 b이다. "TCX"(즉, 변환 코딩 여기를 이용하는 오디오 디코딩)에 대한 디코딩된 계수들 "x_ac_dec[]"은 (예를 들어, 바로) 어레이 "x_tcx_invquant[win][bin]"에 저장되고, 무잡음 코딩 코드워드들의 전송 순서는, 그것들이 어레이에 수신되어 저장된 순서로 디코딩될 때, "bin"이 가장 빠르게 증가하는 인덱스이고 "win"이 가장 느리게 증가하는 인덱스이다. 코드워드 내에서, 디코딩 순서는 a, 그 다음에 b이다.

[0301] 우선, 플래그 "arith_reset_flag"는 콘텍스트가 재설정되어야 하는지를 결정한다. 만약 플래그가 참이라면, 이는 함수 "arith_map_context"에서 고려된다.

[0302] 디코딩 과정은 콘텍스트 성분 벡터 "q"가 "q[1][]"에서 "q[0][]"로 저장된 이전 프레임의 콘텍스트 성분들을 복사하고 맵핑함으로써 업데이트되는 초기화 단계로 시작한다. "q" 내의 콘텍스트 성분들은 2-튜플 당 4 비트에 저장된다. 세부적인 사항들을 위해, 도 5a의 의사 프로그램 코드가 참조된다.

[0303] 무잡음 디코더는 무보호 양자화된 스펙트럼 계수들의 2-튜플들을 출력한다. 처음에, 디코딩하기 위한 2-튜플들에 관련되는 이전에 디코딩된 스펙트럼 계수들에 기초하여 콘텍스트의 상태 c가 계산된다. 그러므로, 상기 상태

는 단지 2개의 새로운 2-튜플들만을 고려하여 마지막에 디코딩된 2-튜플의 콘텍스트 상태를 이용해 증가하여 업데이트된다. 상기 상태는 17 비트로 디코딩되고, 함수 "arith_get_context"에 의해 반환된다. 설정 함수 "arith_get_context"의 의사 프로그램 코드 표현이 도 5c에 도시된다.

[0304] 콘텍스트 상태 c는 최상위 2 비트 방식 평면 m을 디코딩하기 위해 이용되는 누적 빈도 테이블을 결정한다. c로부터 상응하는 누적 빈도 테이블 인덱스 "pki"로의 맵핑은 함수 "arith_get_pk()"에 의해 수행된다. 함수 "arith_get_pk()"에 대한 의사 프로그램 코드 표현이 도 5e에 도시된다.

[0305] 값 m은 누적 빈도 테이블 "arith_cf_m[pki][]"를 갖는 호출된 함수 "arith_decode()"을 이용하여 디코딩되는데, 여기서 "pki"는 "arith_get_pk()"에 의해 반환된 인덱스에 상응한다. 산술 코더(및 디코더)는 스케일링과 함께 태그를 생성하는 방법을 이용하는 정수 구현이다. 도 5g에 따른 의사 프로그램 코드는 이용된 알고리즘을 기술한다.

[0306] 디코딩된 값 m이 이스케이프 심볼 "ARITH_ESCAPE"일 때, 변수들 "lev" 및 "esc_nb"는 1 만큼 증가되고 다른 값 m이 디코딩된다. 이 경우에, 입력 인수로써 값 "c+ esc_nb<<17"을 갖는 함수 "get_pk()"는 다시 한번 호출되는데, 여기서 "esc_nb"는 동일한 2-튜플에 대해 이전에 디코딩되고 7 개로 경계지어진 이스케이프 심볼들의 수이다.

[0307] 일단 값 m이 이스케이프 심볼 "ARITH_ESCAPE"이 아니면, 디코더는 연이은 m이 "ARITH_STOP" 심볼인지를 검사한다. 만약 조건 "(esc_nb>0&&m==0)"이 참이라면, "ARITH_STOP" 심볼이 감지되고 디코딩 과정이 종료된다. 디코더는 이후에 기술된 부호 디코딩으로 바로 점프한다. 상기 조건은 프레임의 나머지가 9 값들로 구성되는 것을 의미한다.

[0308] 만약 "ARITH_STOP"을 충족시키지 않는다면, 잔여 비트 평면들이, 그 다음에, 디코딩되며, 만약 있다면, 현재 2-튜플이 디코딩된다. 잔여 비트 평면들은, 누적 분포 테이블 "arith_cf_r[]"을 갖는 "arith_decode()"을 lev 횟수 호출함으로써, 최상위로부터 최하위 레벨까지 디코딩된다. 디코딩된 비트 평면들 r은 그 의사 프로그램 코드가 도 5j에 도시되는 알고리즘에 따라 이전에 디코딩된 값 m의 정제를 허용한다. 이 시점에서, 2-튜플 (a,b)의 무부호 값이 완전히 디코딩된다. 그것은 그 의사 프로그램 코드 표현이 도 5k에 도시되는 알고리즘에 따라 스펙트럼 계수들을 가지고 있는 성분 내에 보관된다.

[0309] 콘텍스트 "q"는 또한 다음 2-튜플을 위해 업데이트된다. 이 콘텍스트 업데이트는 마지막 2-튜플을 위해서도 수행됨을 알아야 한다. 이 콘텍스트 업데이트는 그 의사 프로그램 코드 표현이 도 5l에 도시되는 함수 "arith_update_context()"에 의해 수행된다.

[0310] 프레임의 다음 2-튜플은, 그 다음에, 1 만큼 증가되고, 함수 "arith_get_context()"로부터 시작하여, 상기와 같이 기술된 동일한 과정을 다시 행함으로써 디코딩된다. 1g/2 2-튜플들이 프레임 내에서 디코딩될 때나, 중지 심볼 "ARITH_STOP"이 발생할 때, 스펙트럼 진폭의 디코딩 과정이 종료되고 부호들의 디코딩이 시작된다.

[0311] 디코딩은 함수 "arith_finish()"을 호출함으로써 종료된다. 잔여 스펙트럼 계수들은 0으로 설정된다. 각각의 콘텍스트 상태들은 그에 상응하여 업데이트된다. 함수 "arith_finish"에 대한 의사 프로그램 코드 표현이 도 5m에 도시된다.

[0312] 일단 모든 무부호 양자화된 스펙트럼 계수들이 디코딩되면, 그에 따른 부호가 추가된다. 각각의 널이 아닌 양자

화된 값 "x_ac_dec"에 대하여, 하나의 비트가 판독된다. 만약 판독된 비트가 0과 같다면, 양자화된 값은 양이고, 아무것도 행해지지 않고, 부호 값은 이전에 디코딩된 무부호 값과 같다. 그렇지 않으면, 디코딩된 계수는 음이고 무부호 값으로부터 2의 보수가 취해진다. 부호 비트는 낮은 주파수들에서 높은 주파수들까지 판독된다.

[0313] 11.12 범례들

[0314] 도 5q는 도 5a, 5c, 5e, 5f, 5g, 5j, 5k, 5l, 및 5m에 따른 알고리즘들에 관련되는 정의들에 대한 범례를 도시한다.

[0315] 도 5r는 도 5b, 5d, 5f, 5h, 5i, 5n, 5o, 및 5p에 따른 알고리즘들에 관련되는 정의들에 대한 범례를 도시한다.

[0316] 12. 맵핑 테이블들

[0317] 본 발명에 따른 일 실시예에서, 특히 유리한 테이블들 "ari_lookup_m", "ari_hash_m", 및 "ari_cf_m"이 도 5e 또는 도 5f에 따른 함수 "arith_get_pk()"의 실행, 및 도 5g, 5h, 및 5i을 참고하여 논의된 함수 "arith_decode()"의 실행을 위해 이용된다. 그러나, 본 발명에 따른 몇몇 실시예들에서 각각 다른 테이블들이 이용될 수 있음을 알아야 한다.

[0318] 12.1 도 22에 따른 테이블 "ari_hash_m[600]"

[0319] 그 제1 실시예가 도 5e를 참조하여 기술되고 그 제2 실시예가 도 5f를 참조하여 기술된 함수 "arith_get_pk"에 의해 이용되는 테이블 "arith_get_pk"에 대한 특히 유리한 구현의 콘텐츠가 도 22에 도시된다. 도 22의 테이블은 테이블 (또는 어레이) "ari_hash_m[600]"의 600 개의 엔트리들을 열거한다는 것을 알아야 한다. 또한, 도 22의 테이블 표현은 성분 인덱스들의 순서로 성분들을 보여주어, 제1 값 "0x000000100UL"은 성분 인덱스 (또는 테이블 인덱스) 0을 갖는 테이블 엔트리 "ari_hash_m[0]"에 상응하고, 마지막 값 "0x7ffffffff4fUL"은 성분 인덱스 또는 테이블 인덱스 599를 갖는 테이블 엔트리 "ari_hash_m[599]"에 상응한다는 것을 또한 알아야 한다. 여기서, "0x"는 테이블 "ari_hash_m[]"의 테이블 엔트리들이 16진 형식으로 표현되는 것을 가리킴을 추가로 알아야 한다. 또한, 여기서, 접미사 "UL"은 테이블 "ari_hash_m[]"의 테이블 엔트리들이 (32 비트의 정확도를 갖는) 무부호 "long" 정수 값들로 표현되는 것을 가리킴을 알아야 한다.

[0320] 뿐만 아니라, 도 22에 따른 테이블 "ari_hash_m[]"의 테이블 엔트리들은, 함수 "arith_get_pk()"의 테이블 검색(506b, 508b, 510b)의 실행을 가능하게 하기 위해, 수치적 순서로 배열됨을 알아야 한다.

[0321] 테이블 "ari_hash_m"의 테이블 엔트리들의 최상위 24 비트는 특정 유효 상태 값들을 표현하며, 한편 최하위 8 비트는 맵핑 규칙 인덱스 값들 "pki"를 표현함을 추가로 알아야 한다. 그러므로, 테이블 "ari_hash_m[]"의 엔트리들은 맵핑 규칙 인덱스 값 "pki"로의 콘텍스트 값의 "직접 히트(direct hit)" 맵핑을 기술한다.

[0322] 그러나, 테이블 "ari_hash_m[]"의 엔트리들의 최상위 24 비트는, 동시에, 동일한 맵핑 규칙 인덱스 값이 연관되는 수치적 콘텍스트 값들의 구간들의 구간 경계들을 표현한다. 이러한 구상에 관한 세부사항들은 이미 상기에서 논의되었다.

[0323] 12.2 도 21에 따른 테이블 "ari_lookup_m"

[0324] 테이블 "ari_lookup_m"에 대한 특히 유리한 실시예의 콘텐츠가 도 21의 테이블에서 도시된다. 여기서, 도 21의 테이블은 테이블 "ari_lookup_m"의 엔트리들을 열거함을 알아야 한다. 상기 엔트리들은, 예를 들어, "i_max" 또는 "i_min"으로 지칭되는 ("성분 인덱스" 또는 "어레이 인덱스" 또는 "테이블 인덱스"라고도 지칭되는) 1차 정수형 엔트리 인덱스에 의해 참조된다. 총 600 개의 엔트리들을 포함하는 테이블 "ari_lookup_m"은 도 5e 또는 5f에 따른 함수 "arith_get_pk"에 의해 이용되기에 매우 적합하다는 것을 알아야 한다. 도 21에 따른 테이블 "ari_lookup_m"은 도 22에 따른 테이블 "ari_hash_m"과 협력하도록 적응됨을 또한 알아야 한다.

[0325] 테이블 "ari_lookup_m[600]"의 엔트리들은 0과 599 사이의 테이블 인덱스 "i"(즉, "i_min" 또는 "i_max")의 오름차순으로 열겨됨을 알아야 한다. 용어 "0x"는 테이블 엔트리들이 16진 형식으로 기술됨을 가리킨다. 이에 따라, 제1 테이블 엔트리 "0x02"는 테이블 인덱스 0을 갖는 테이블 엔트리 "ari_lookup_m[0]"에 상응하고, 마지막 테이블 엔트리 "0x5E"는 테이블 인덱스 599를 갖는 테이블 엔트리 "ari_lookup_m[599]"에 상응한다.

[0326] 테이블 "ari_lookup_m[]"의 엔트리들은 테이블 "arith_hash_m[]"의 인접한 엔트리들에 의해 정의된 구간들과 연관된다는 것을 또한 알아야 한다. 그러므로, 테이블 "ari_lookup_m"의 엔트리들은 수치적 콘텍스트 값들의 구간들과 연관된 맵핑 규칙 인덱스 값들을 기술하는데, 여기서 구간들은 테이블 "arith_hash_m"의 엔트리들에 의해 정의된다.

[0327] 12.3 도 23에 따른 테이블 "ari_cf_m[96][17]"

[0328] 도 23은 96 개의 누적 빈도 테이블들 (또는 서브 테이블들) "ari_cf_m[pki][17]"의 셋트를 도시하는데, 그 중의 하나가, 예를 들어, 함수 "arith_decode()"의 실행을 위해, 즉, 최상위 비트 평면 값의 디코딩을 위해, 오디오 인코더(100, 700) 또는 오디오 디코더(200, 800)에 의해 선택된다. 도 23에 도시된 96 개의 누적 빈도 테이블들 (또는 서브 테이블들) 중에 선택된 하나는 함수 "arith_decode()"의 실행에서 테이블 "cum_freq[]"의 기능을 한다.

[0329] 도 23에서 알 수 있는 바와 같이, 각각의 서브 블록은 17 개의 엔트리들을 갖는 누적 빈도 테이블을 표현한다. 예를 들어, 제1 서브 블록(2310)은 "pki=0"에 대한 누적 빈도 테이블의 17 개의 엔트리들을 표현한다. 제2 블록(2312)은 "pki=1"에 대한 누적 빈도 테이블의 17 개의 엔트리들을 표현한다. 마지막으로, 96번째 서브 블록(2396)은 "pki=95"에 대한 누적 빈도 테이블의 17 개의 엔트리들을 표현한다. 그러므로, 도 23은 "pki=0" 내지 "pki=95"에 대한 96 개의 각각 다른 누적 빈도 테이블들(또는 서브 테이블들)을 효과적으로 표현하는데, 여기서 각가가의 96 개의 누적 빈도 테이블들은 (굽은 괄호들의 의해 둘러싸인) 서브 블록에 의해 표현되고, 여기서 각각의 상기 누적 빈도 테이블들은 17 개의 엔트리들을 포함한다.

[0330] 서브 블록(예를 들어, 서브 블록 2310 또는 2312, 또는 서브 블록 2396) 내에서, 제1 값은 (어레이 인덱스 또는 테이블 인덱스 0을 갖는) 누적 빈도 테이블의 제1 엔트리를 기술하고, 마지막 값은 (어레이 인덱스 또는 테이블 인덱스 16을 갖는) 누적 빈도 테이블의 마지막 엔트리를 기술한다.

[0331] 이에 따라, 도 23의 테이블 표현의 각각의 서브 블록(2310, 2312, 2396)은 도 5g에 따른, 또는 도 5h 및 5i에 따른 함수 "arith_decode"에 의해 이용되기 위한 누적 빈도 테이블의 엔트리들을 표현한다. 함수 "arith_decode"의 입력 변수 "cum_freq[]"는 (테이블 "arith_cf_m"의 17 개의 엔트리들의 개개의 서브 브록들에 의해 표현된) 96 개의 누적 빈도 테이블들 중 어느 것이 현재 스펙트럼 계수들의 디코딩에 이용되어야 하는지를 기술한다.

[0332] 12.4 도 24에 따른 테이블 "ari_cf_r[]"

[0333] 도 24는 테이블 "ari_cf_r[]"의 콘텐츠를 도시한다.

[0334] 상기 테이블의 4 개의 엔트리들이 도 24에 도시된다. 그러나, 테이블 "ari_cf_r"은 다른 실시예들에서는 결국 각각 다를 수 있음을 알아야 한다.

[0335] 13. 성능 평가 및 장점

[0336] 본 발명에 따른 실시예들은, 계산 복잡도, 메모리 요구, 및 코딩 효율성 사이의 개선된 균형을 획득하기 위해, 상기에서 논의된 바와 같이, 업데이트된 함수들 (또는 알고리즘들) 및 업데이트된 테이블들의 셋트를 이용한다.

[0337] 일반적으로 말하면, 본 발명에 따른 실시예들은 개선된 스펙트럼 무잡음 코딩을 창출한다. 본 발명에 따른 실시예들은 USAC(통합 음성 오디오 인코딩)에서의 스펙트럼 무잡음 코딩에 대한 향상을 기술한다.

[0338] 본 발명에 따른 실시예들은, MPEC 제안 논문들(input papers, m16912 및 m17002)에서 제시된 기법들에 기초하여, 스펙트럼 계수들에 대한 개선된 스펙트럼 무잡음 코딩에 관하여 CE에 관한 업데이트된 제안을 창출한다. 제안들 모두가 평가되었으며, 잠재적 결점들은 제거되었고 강점들은 결합되었다.

[0339] m16912 및 m17002에서, 결의 제안(resulting proposal)은 규격 초안 5 USAC(통합 음성 오디오 코딩에 관한 표준 초안)와 같이 원래(original) 콘텍스트 기반 산술 코딩 기법에 기초하고 있으나, 계산 복잡도를 증가시키지 않으면서 메모리 요구(랜덤 엑세스 메모리(random access memory, RAM), 및 읽기 전용 메모리(read-only memory, ROM))를 상당히 줄일 수 있으며, 한편 코딩 효율성을 유지한다. 또한, USAC 표준 초안의 규격 초안 3 및 USAC 표준 초안의 규격 초안 5에 따른 비트스트림들에 대한 무손실 트랜스코딩(transcoding)이 가능한 것으로 증명되었다. 본 발명에 따른 실시예들은 USAC 표준 초안의 규격 초안 5에서 이용된 스펙트럼 무손실 코딩 기법을 대체하는 것을 목표로 한다.

[0340] 여기서 기술된 산술 코딩 기법은 USAC 표준 초안의 참조 모델 0(RM0) 또는 규격 초안 5(WD)에서와 같은 기법에 기초한다. 주파수 또는 시간에서의 스펙트럼 계수들은 콘텍스트를 모델링한다. 이러한 콘텍스트는 산술 인코더를 위한 누적 빈도 테이블들의 선택에 이용된다. 규격 초안 5(WD)와 비교하여, 콘텍스트 모델링이 더 개선되고 심볼 확률을 가지고 있는 테이블들이 리트레이닝(re-train) 되었다. 각각 다른 확률 모델들의 수가 32에서 96까지 증가되었다.

[0341] 본 발명에 따른 실시에들은 테이블 크기들(데이터 ROM 요구)를 32 비트 길이의 1518 워드 또는 6072 바이트(WD: 16,894.5 워드 또는 67,578 바이트)로 감소시킨다. 정적 RAM 요구는 코어 코더 채널 당 666 워드(2,664 바이트)에서 72 워드(288 바이트)로 감소된다. 동시에, 코딩 성능을 충분히 보존하고, 심지어 모든 9 개의 동작점들에 걸쳐 전체 데이터율과 비교하여 대략 1.29 내지 1.95%의 이득에 도달할 수 있다. 모든 규격 초안 3 및 규격 초안 5의 비트스트림들은, 비트 보유 제약에 영향을 미치지 않으면서, 무손실 방식으로 트랜스코딩될 수 있다.

[0342] 다음에서는, 여기서 기술된 구상의 장점들에 대한 이해를 용이하게 하기 위해 USAC 표준 초안의 규격 초안 5에 따른 코딩 구상들에 대한 간략한 논의가 제공될 것이다. 이어서, 본 발명에 따른 몇몇 바람직한 실시예들이 기

술될 것이다.

[0343] USAC 규격 초안 5에서, 콘텍스트 기반 산술 코딩 기법은 양자화된 스펙트럼 계수들의 무손실 코딩을 위해 이용된다. 주파수와 시간에서 앞서는 콘텍스트에 따라, 디코딩된 스펙트럼 계수들이 이용된다. 규격 초안 5에서, 최대 16 개의 스펙트럼 값들의 수가, 그 중 12 개가 시간에서 앞서는, 콘텍스트로 이용된다. 또한, 콘텍스트를 위해 이용되고 디코딩되는 스펙트럼 계수들은 4-튜플들(즉, 주파수에서 근처에 있는 4 개의 스펙트럼 계수들, 도 14a 참조)로 그룹을 이루게 된다. 콘텍스트는 감소되어 누적 빈도 테이블에 맵핑되며, 이는, 그 다음에, 스펙트럼 계수들의 다음 4-튜플을 디코딩하기 위해 이용된다.

[0344] 완전한(complete) 규격 초안 5의 무잡음 코딩 기법에 있어서, 16894.5 워드(67578 바이트)의 메모리(읽기 전용 메모리(ROM)) 수요가 요구된다. 또한, 다음 프레임을 위해 상태들을 저장하기 위해 코어 코더 채널 당 정적 RAM의 666 워드(2664 바이트)가 요구된다. 도 14b의 테이블 표현은 USAC WD4 산술 코딩 기법에서 이용되는 테이블들을 기술한다.

[0345] 여기서, 무잡음 코딩에 관해서, USAC 표준 초안의 규격 초안들 4 및 5는 동일하다. 둘 다 동일한 무잡음 코더를 이용한다.

[0346] 완전한 USAC WD5 디코더의 전체 메모리 요구는 프로그램 코드가 없는 데이터 ROM에 대해 37000 워드(148000 바이트), 그리고 정적 RAM에 대해 10000 내지 17000 워드가 될 것으로 추정된다. 무잡음 코더 테이블들이 전체 데이터 ROM 요구의 대략 45%를 소모함을 명확히 알 수 있다. 가장 큰 개별 테이블은 이미 4096 워드(16384 바이트)를 소모한다.

[0347] 모든 테이블들의 결합과 큰 개별 테이블들의 크기는 모두, 일반적으로 8 내지 32 킬로바이트(예를 들어, ARM9e, TIC64XX, 등) 범위 내에 있는 소비자 휴대용 장치들에서 이용되는 고정 소수점 처리기들에 의해 제공되는 것과 같은 캐쉬 크기를 일반적으로 초과하는 것으로 확인됐다. 이는 테이블들의 셋트가 데이터로의 빠른 랜덤 액세스를 가능하게 하는 고속 데이터 RAM에 아마도 저장될 수 없음을 의미한다. 이는 전체 디코딩 과정이 느려지도록 한다.

[0348] 또한, HE-AAC와 같은 현재의 성공적인 오디오 코딩 기술은 대부분의 이동 장치들에서 구현가능함이 증명된 것으로 확인됐다. HE-AAC는 995 워드의 테이블 크기를 갖는 허프만(Huffman) 엔트로피 코딩 기법을 이용한다. 세부적인 사항들을 위해, ISO/IEC JTC1/SC29/WG11 N2005, MPEG98, 1998년 2월, 산호세, "MPEG-2 AAC2의 복잡도에 관한 수정 보고서(Revised Report on Complexity of MPEG-2 AAC2)"가 참조된다.

[0349] 제90회 MPEG 회의에서, MPEG 제안 논문들 m16912 및 m17002에, 메모리 요구 감소 및 무잡음 코딩 기법의 인코딩 효율성 개선을 목표로 하는 2 가지 제안들이 제시되었다. 두 제안들을 분석함으로써, 다음의 결론이 얻어질 수 있다.

[0350] ● 코드 워드의 크기를 줄임으로써 메모리 요구에 대한 상당한 감소가 가능하다. MPEG 제안 문서 m17002에서 제시된 바와 같이, 4-튜플들에서 1-튜플들로 크기를 줄임으로써, 코딩 효율을 침해하지 않으면서 16984.5에서 900 워드로 메모리 요구가 감소될 수 있고;

[0351] ● 균일 확률 분포를 이용하는 대신에, LSB 코딩을 위해 비 균일 확률 분포의 코드 북을 적용함으로써 추가적인 중복이 제거될 수 있다.

[0352] 이러한 평가 과정에서, 4-튜플에서 1-튜플로 바꾸는 코딩 기법은 계산 복잡도에 상당한 영향을 미치는 것으로 확인되었는데: 코딩 크기의 감소는 코딩하기 위한 심볼들의 수를 동일한 인자로 증가시킨다. 이는, 4-튜플들에서 1-튜플로의 감소를 위해, 콘텍스트를 결정하며, 해쉬 테이블들에 접근하고, 심볼들을 디코딩하기 위해 필요한 연산들이 이전보다 4 배 더 자주 수행되어야 함을 의미한다. 콘텍스트 결정을 위한 좀더 복잡한 알고리즘과 함께, 이는 인자 2.5 또는 x.xxPCU 만큼 계산 복잡도에서의 증가를 초래한다.

[0353] 다음에서는, 본 발명의 실시예들에 따른 제안된 새로운 기법이 간략히 기술될 것이다.

[0354] 메모리 풋프린트(footprint) 및 계산 복잡도의 문제를 극복하기 위해, 규격 초안 5(WD5)에서의 기법을 대체하도록 개선된 무잡음 코딩 기법이 제안된다. 메모리 요구를 줄이는 한편, 압축 효율성을 유지하고, 계산 복잡도를 증가시키지 않는데 개발의 주요 초점을 뒀다. 좀더 구체적으로, 목표는 압축 수행의 다차원 복잡도 공간, 복잡도, 및 메모리 요구에서 훌륭한 (또는 심지어 최상의) 균형에 도달하는 것이었다.

[0355] 새로운 코딩 기법 제안은 WD5 무잡음 인코더의 주요 특징, 즉, 콘텍스트 적응을 차용한다. 콘텍스트는 WD5에서 과거 및 현재 프레임(여기서, 프레임은 오디오 콘텐츠의 일부분으로 여겨질 수 있다) 모두로부터 나오는 이전에 디코딩된 스펙트럼 계수들을 이용하여 도출된다. 그러나, 스펙트럼 계수들은, 이제, 2-튜플을 형성하기 위해 2개의 계수들을 함께 결합함으로써 코딩된다. 다른 차이는 스펙트럼 계수들이, 이제, 부호, 좀더 유효한 비트들 또는 최상위 비트들(MSBs), 및 하위 비트들 또는 최하위 비트들(LBSs)의 세 부분으로 나누어진다는 사실에 있다. 부호는 크기와 관계없이 코딩되는데, 이는, 만약 존재한다면, 최상위 비트들(또는 좀더 유효한 비트들) 및 상기 비트들의 나머지(또는 하위 비트들)의 두 부분으로 추가로 나누어진다. 두 성분들의 크기가 3보다 낮거나 3과 같은 2-튜플들은 MSB 코딩에 의해 바로 코딩된다. 그렇지 않으면, 임의의 추가적인 비트 평면을 신호로 알리기 위해 우선 이스케이프 코드워드가 전송된다. 기본 버전에서는, 누락된 정보, LSB, 및 부호가 균일 확률 분포를 이용하여 모두 코딩된다. 그렇지 않으면, 각각 다른 확률 분포가 이용될 수 있다.

[0356] 테이블 크기 감소는 여전히 가능한데, 왜냐하면:

[0357] ● 단지 17 개의 심볼들에 대한 확률만이 저장될 필요가 있으며; {[0;+3], [0;+3]}+ESC 심볼;

[0358] ● 그룹을 이루는 테이블(egroups, dgroups, dgvectors)을 저장할 필요가 없고;

[0359] ● 해시 테이블의 크기는 적절한 트레이닝으로 감소될 수 있기 때문이다.

[0360] 다음에서는, MSB 코딩에 관한 몇몇 세부사항들이 기술될 것이다. 이미 언급한 바와 같이, USAC 표준 초안의 WD5, 제90회 MPEG 회의에서 제출된 제안, 및 현재 제안 사이의 주요 차이들 중 하나는 심볼들의 크기이다. USAC 표준 초안의 WD5에서, 콘텍스트 생성 및 무잡음 코딩을 위해 4-튜플들이 고려되었다. 제90회 MPEG 회의에서 제출된 제안에서는, ROM 요구를 감소시키기 위해 대신에 1-튜플들이 이용되었다. 개발 과정에서, 계산 복잡도를 증가시키지 않으면서, ROM 요구를 감소시키기 위해 2-튜플들이 최고의 절충이 되는 것으로 확인됐다. 콘텍스트 혁신을 위해 4 개의 4-튜플들을 고려하는 대신에, 이제 4 개의 2-튜플들이 고려된다. 도 15a에 도시된 바와 같이, 3 개의 2-튜플들은 (오디오 콘텐츠의 이전 부분으로 지칭되기도 하는) 과거 프레임으로부터 나오고, 하나는 (오디오 콘텐츠의 현재 부분으로 지칭되기도 하는) 현재 프레임으로부터 나온다.

[0361] 테이블 크기 감소는 3 가지 주요 요인들에서 기인한다. 우선, 단지 17 W개의 심볼들에 대한 확률만이 저장될

필요가 있다(즉, {[0;+3], [0;+3]} + ESC 심볼). 그룹을 이루는 테이블들(즉, egroups, dgroups, 및 dgvectors)은 더 이상 요구되지 않는다. 마지막으로, 해시 테이블의 크기가 적절한 트레이닝을 수행함으로써 감소됐다.

[0362] 비록 크기가 4에서 2로 감소되었지만, 복잡도는 USAC 표준 초안의 WD5에서와 같은 범위로 유지되었다. 이는 콘텍스트 생성 및 해시 테이블 접근 모두를 간소화함으로써 달성되었다.

[0363] 코딩 성능이 영향을 받지 않고, 심지어 약간 개선된 방식으로 각각 다른 간소화 및 최적화가 행해졌다. 주로 32에서 96으로 확률 모델들의 수를 증가시킴으로써 달성되었다.

[0364] 다음에서는, LSB 코딩에 관한 몇몇 세부사항들이 기술될 것이다. LSB는 몇몇 실시예들에서 균일 확률 분포로 코딩된다. USAC의 WD5와 비교하여, LSB는 4-튜플들 대신에 2-튜플들 이내로 고려된다.

[0365] 다음에서 부호 코딩에 관한 몇몇 세부사항들이 설명될 것이다. 부호는 복잡도 감소를 위해 산술 코어 코더를 이용하지 않고 코딩된다. 오직 상응하는 크기가 널이 아닐 때에만, 부호는 1 비트로 전송된다. 0은 양의 값을 의미하고 1은 음의 값을 의미한다.

[0366] 다음에서는, 메모리 요구에 관한 몇몇 세부사항들이 설명될 것이다. 제안된 새로운 기법은 기껏해야 1522.5의 새로운 워드(6090 바이트)의 전체 ROM 요구를 제시한다. 세부적인 사항들을 위해, 제안된 코딩 기법에서 이용되는 것과 같은 테이블들을 기술하는 도 15b의 테이블이 참조된다. USAC 표준 초안의 WD5에서의 무잡음 코딩 기법의 ROM 요구와 비교하여, ROM 요구는 적어도 15462 워드(61848 바이트)로 감소된다. 이제, 결국 HE-AAC(995 워드 또는 3980 바이트)에서 AAC 허프만 디코더에 필요로 하는 메모리 요구와 같은 크기의 요구를 하게 된다. 세부적인 사항들을 위해, ISO/IEC JTC1/SC29/WG11 N2005, MPEG98, 1998년 2월, San Jos, "MPEG-2 AAC2의 복잡도에 관한 수정 보고서", 및 또한 도 16a가 참조된다. 이는 무잡음 코더의 전체 ROM 요구를 92% 이상, 그리고 완전한 USAC 디코더의 전체 ROM 요구를 대략 37000 워드에서 대략 21500 워드로, 또는 41% 이상으로 감소시킨다. 세부적인 사항들을 위해, 도 16a 및 16b가 다시 참조되는데, 여기서 도 16a는 제안된 바와 같은 무잡음 코딩 기법, 및 USAC 표준 초안의 WD4에 따른 무잡음 코딩 기법의 ROM 요구를 도시하고, 여기서 도 16b는 제안된 기법 및 USAC 표준 초안의 WD4에 따른 전체 USAC 디코더 데이터 ROM 요구를 도시한다.

[0367] 계속하여, 다음 프레임(정적 ROM)에서 콘텍스트 도출을 위해 요구되는 정보의 양이 또한 감소된다. USAC 표준 초안의 WD5에서, 분해능 10 비트의 4-튜플 당 하나의 그룹 인덱스에 추가해 일반적으로 16 비트의 분해능을 갖는 계수들(최대 1152 개의 계수들)의 완전한 셋트가 저장되어야 했는데, 이는 통산하면 코어 코더 채널 당 666 워드(2664 바이트)이다(완전한 USAC WD4 디코더: 대략 10000 내지 17000 워드). 새로운 기법은 반복되는 정보를 스펙트럼 계수당 단지 2 비트로 감소시키는데, 이는 통산하면 코어 코더 채널 당 전체적으로 72 워드(288 비트)이다. 정적 메모리에 대한 요구가 594 워드(2376 바이트)로 감소될 수 있다.

[0368] 다음에서는, 코딩 효율성의 증가 가능성에 관한 몇몇 세부사항들이 기술될 것이다. 새로운 제안에 따른 실시예들의 디코딩 효율성이 USAC 표준 초안 규규격 초안 3(WD3) 및 WD5에 따른 기준 품질 비트스트림들과 비교되었다. 상기 비교는 기준 소프트웨어 디코더에 기초하여 트랜스코더로 수행되었다. USAC 표준 초안의 WD3 또는 WD5에 따른 무잡음 코딩과 제안된 코딩 기법의 상기 비교에 관한 세부적인 사항들을 위해, 제안된 코딩 기법과 WD3/5 무잡음 코딩의 비교를 위한 테스트 준비에 대한 도식적인 표현을 도시하는 도 17이 참조된다.

[0369] 또한, 본 발명에 따른 실시예에서의 메모리 요구가 USAC 표준 초안의 WD3(또는 WD5)에 따른 실시예들에 비교되었다.

[0370] 코딩 효율성이 유지되었을 뿐만 아니라, 약간 증가된다. 세부적인 사항들을 위해, WD3 산술 코더(또는 WD3 산술 코더를 이용하는 USAC 오디오 코더), 및 본 발명의 일 실시예에 따른 오디오 코더(예를 들어, USAC 오디오 코더)에 의해 생기는 평균 비트율에 대한 테이블 표현이 도시되는 도 18의 테이블이 참조된다.

[0371] 연산 모드 당 평균 비트율에 대한 세부사항들은 도 18의 테이블에서 확인될 수 있다.

[0372] 또한, 도 19는 WD3 산술 코더(또는 WD3 산술 코더를 이용하는 오디오 코더) 및 본 발명의 일 실시예에 따른 오디오 코더에 대한 최소 및 최대 비트 저장 레벨의 테이블 표현을 도시한다.

[0373] 다음에서는, 계산 복잡도에 관한 몇몇 세부사항들이 기술될 것이다. 산술 코딩의 차원수의 감소는 일반적으로 계산 복잡도의 증가를 불러온다. 사실, 인자 2로 차원을 감소시키는 것은 산술 코더 루틴들을 2번 호출하게 만들 것이다.

[0374] 그러나, 이러한 복잡도의 증가는 본 발명의 실시예들에 따라 제안된 새로운 코딩 기법에서 소개된 여러 최적화에 의해 제한될 수 있는 것으로 확인됐다. 콘텍스트 생성은 본 발명에 따른 몇몇 실시예들에서 크게 간소화된다. 각각의 2-튜플에 대해, 콘텍스트는 마지막에 발생된 콘텍스트로부터 증가하여 업데이트될 수 있다. 확률이, 이제, 16 비트 대신 14 비트로 저장되는데, 이는 디코딩 과정 중에 64 비트 연산을 피한다. 또한, 본 발명에 따른 몇몇 실시예들에서 확률 모델 맵핑이 매우 최적화된다. 최악의 경우 대폭적으로 감소되었고 95 회 대신 10 회 반복으로 제한된다.

[0375] 결과적으로, 제안된 무잡음 코딩 기법의 계산 복잡도는 WD 5에서와 동일한 범위로 유지되었다. 무잡음 코딩의 각각 다른 버전들에 의해 "펜과 종이" 측정이 수행되었고 도 20의 테이블에 기록된다. 새로운 코딩 기법이 WD5 산술 코더보다 단지 약 13% 덜 복잡함을 보여준다.

[0376] 상기를 요약하면, 본 발명에 따른 실시예들은 계산 복잡도, 메모리 요구, 및 코딩 효율성 사이의 특히 좋은 균형을 제공함을 알 수 있다.

[0377] 14. 비트스트림 구문

[0378] 14.1 스펙트럼 무잡음 코더의 페이로드

[0379] 다음에서는, 스펙트럼 무잡음 코더의 페이로드들에 관한 몇몇 세부사항들이 기술될 것이다. 몇몇 실시예들에서, 예를 들어, 이른바 "선형 예측 도메인" 코딩 모드 및 "주파수 도메인" 코딩 모드와 같은 복수의 각각 다른 코딩 모드들이 있다. 선형 예측 도메인 코딩 모드에서, 오디오 신호의 선형 예측 분석에 기초하여 잡음 정형(noise shaping)이 수행되고, 잡음 정형된 신호는 주파수 도메인에서 인코딩된다. 주파수 도메인 코딩 모드에서, 심리음향적 분석에 기초하여 잡음 정형이 수행되고, 오디오 콘텐츠의 잡음 정형된 버전은 주파수 도메인에서 인코딩된다.

[0380] "선형 예측 도메인" 코딩된 신호 및 "주파수 도메인" 코딩된 신호 모두로부터의 스펙트럼 계수들은 스칼라(scalar) 양자화되고, 그 다음에, 적응된 콘텐츠에 따르는 산술 코딩에 의해 무잡음 코딩된다. 양자화된 계수들은 가장 낮은 주파수에서 가장 높은 주파수로 전송되기 전에 2-튜플들로 함께 모인다. 각각의 2-튜플들은 (만약

에 있다면) 부호 s, 최상위 2 비트 방식 평면 m, 및 하나 이상의 잔여 하위 비트 평면들 r로 나누어진다. 값 m은 근처에 있는 스펙트럼 계수들에 의해 정의된 콘텍스트에 따라 코딩된다. 다시 말해서, m은 근처의 계수들에 따라 코딩된다. 잔여 하위 비트 평면들 r은 콘텍스트를 고려하지 않고 엔트로피(entropy) 코딩된다. m 및 r에 의해, 이러한 스펙트럼 계수들의 진폭이 디코더 측에서 복원될 수 있다. 모든 널이 아닌 심볼들을 위해, 1 비트를 이용하여 산술 코더의 외부에서 부호 s가 코딩된다. 다시 말해서, 값 m과 r은 산술 코더의 심볼들을 형성한다. 마지막으로, 부호 s는 널이 아닌 양자화된 계수 당 1 비트를 이용하의 산술 코더의 외부에서 코딩된다.

[0381] 상세한 산술 코딩 절차가 여기서 기술된다.

[0382] 14.2 구문 성분들

[0383] 다음에서는, 산술적으로 인코딩된 스펙트럼 정보를 전달하는 비트스트림의 비트스트림 구문이 도 6a 및 6j를 참조하여 기술될 것이다.

[0384] 도 6a는 이른바 USAC 원시 데이터(raw data) 블록("usac_raw_data_block()")의 구문 표현을 도시한다.

[0385] USAC 원시 데이터 블록은 하나 이상의 단일 채널 성분들("ingle_channel_element()") 및/또는 하나 이상의 채널 쌍 성분들("channel_pair_element()")을 포함한다.

[0386] 이제 도 6b를 참조하면, 단일 채널 성분의 구문이 기술된다. 단일 채널 성분은 코어 모드에 따라 선형 예측 도메인 채널 스트림("lpd_channel_stream ()") 또는 주파수 도메인 채널 스트림("fd_channel_stream ()")을 포함한다.

[0387] 도 6c는 채널 쌍 성분의 구문 표현을 도시한다. 채널 쌍 성분은 코어 모드 정보("core_mode0", "core_mode1")를 포함한다. 또한, 채널 쌍 성분은 구성 정보 "ics_info()"를 포함할 수 있다. 부가적으로, 코어 모드 정보에 따라, 채널 쌍 성분은 채널들 중 첫 번째와 연관된 선형 예측 도메인 채널 스트림 또는 주파수 도메인 채널 스트림을 포함하고, 채널 쌍 성분은 또한 채널들 중 두 번째와 연관된 선형 예측 도메인 채널 스트림 또는 주파수 도메인 채널 스트림을 포함한다.

[0388] 그 구문 표현이 도 6d에 도시되는 구성 정보 "ics_info()"는, 본 발명에 대해 특별한 관련이 없는, 복수의 각각 다른 구성 정보 항목들을 포함한다.

[0389] 그 구문 표현이 도 6e에 도시되는 주파수 도메인 채널 스트림("fd_channel_stream ()")은 이득 정보("global_gain") 및 구성 정보("ics_info ()")를 포함한다. 또한, 주파수 도메인 채널 스트림은 각각 다른 스케일링 인자 대역들의 스펙트럼 값들에 대한 스케일링을 위해 이용되는 스케일링 인자들을 기술하고, 예를 들어, 스케일러(150) 및 재스케일러(240)에 의해 적용되는 스케일링 인자 데이터("scale_factor_data ()")를 포함한다. 주파수 도메인 채널 스트림은 또한 산술적으로 인코딩된 스펙트럼 값들을 표현하는 산술적으로 코딩된 스펙트럼 데이터("ac_spectral_data ()")를 포함한다.

[0390] 그 구문 표현이 도 6f에 도시되는 산술적으로 코딩된 스펙트럼 데이터("ac_spectral_data()")는, 상기에서 기술된 바와 같이, 콘텍스트를 선택적으로 재설정하기 위해 이용되는 선택적 산술 재설정 플래그("arith_reset_flag")를 포함한다. 또한, 산술적으로 코딩된 스펙트럼 데이터는 산술적으로 코딩된 스펙트럼 값

들을 전달하는 복수의 산술 데이터 블록들("arith_data")을 포함한다. 산술적으로 코딩된 데이터 블록들의 구조는, 다음에서 논의될 것으로, (변수 "num_bands"에 의해 표현된) 주파수 대역들의 수 및 또한 산술 재설정 플래그의 상태에 따라 결정된다.

[0391] 다음에서는, 상기 산술적으로 코딩된 데이터 블록들의 구문 표현을 도시하는 도 6g를 참조하여 산술적으로 인코딩된 데이터 블록의 구조가 기술될 것이다. 산술적으로 코딩된 데이터 블록 내의 데이터 표현은 인코딩되는 스펙트럼 값들의 수 lg, 산술 재설정 플래그의 상태, 및 또한 콘텍스트, 즉, 이전에 인코딩된 스펙트럼 값들에 따라 결정된다.

[0392] 스펙트럼 값들의 현재 셋트(예를 들어, 2-튜플)의 인코딩하기 위한 콘텍스트는 도면 부호 660에서 도시된 콘텍스트 결정 알고리즘에 따라 결정된다. 콘텍스트 결정 알고리즘에 관한 세부사항들이, 도 5a 및 5b를 참조하여, 상기에서 설명되었다. 산술적으로 인코딩된 데이터 블록은 lg/2 개의 코드워드들의 셋트들을 포함하는데, 각각의 코드워드들의 셋트는 복수의(예를 들어, 2-튜플) 스펙트럼 값들을 표현한다. 코드워드들의 셋트는 1과 20 비트 사이를 이용하여 스펙트럼 값들의 튜플의 최상의 비트 평면 값 m을 표현하는 산술 코드워드 "acod_m[pki][m]"를 포함한다. 또한, 만약 스펙트럼 값들의 튜플이 정확한 표현을 위해 최상위 비트 평면들보다 더 많은 비트 평면들을 요구하면, 코드워드들의 셋트는 하나 이상의 코드워드들 "acod_r[r]"을 포함한다. 코드워드 "acod_r[r]"은 1과 14 비트 사이를 이용하여 하위 비트 평면을 표현한다.

[0393] 만약, 그러나, 스펙트럼 값들의 적절한 표현을 위해 (최상위 비트 평면에 더해) 하나 이상의 하위 비트 평면들이 요구된다면, 이는 하나 이상의 산술 이스케이프 코드워드("ARITH_ESCAPE")를 이용하여 신호로 알려진다. 그러므로, 일반적으로, 스펙트럼 값에 대해, 얼마나 많은 비트 평면들(최상위 비트 평면 및, 어쩌면, 하나 이상의 추가적인 하위 비트 평면들)이 요구되지는 결정된다고 할 수 있다. 만약 하나 이상의 하위 비트 평면들이 요구된다면, 그 누적 빈도 테이블 인덱스가 변수 "pki"에 의해 주어지는 현재 선택된 누적 빈도 테이블에 따라 인코딩되는 하나 이상의 산술 이스케이프 코드워드들 "acod_m[pki][ARITH_ESCAPE]"에 의해 신호로 알려진다. 또한, 만약 하나 이상의 산술 이스케이프 코드워드들이 비트스트림에 포함된다면, 도면 부호들 664, 662에서 알 수 있는 바와 같이, 콘텍스트가 조정된다. 하나 이상의 산술 이스케이프 코드워드들 다음에, 도면 부호 663에 도시된 바와 같이, 산술 코드워드 "acod_m[pki][m]"이 비트스트림에 포함되는데, 여기서 "pki"는 (산술 이스케이프 코드워드들을 포함함으로써 야기되는 콘텍스트 적응을 고려하여) 현재 유효한 확률 모델 인덱스를 지칭하고, m은 인코딩되거나 디코딩되는 스펙트럼 값의 최상위 비트 평면 값을 지칭한다(여기서 m은 "ARITH_ESCAPE" 코드워드와 다르다.).

[0394] 상기에서 논의된 바와 같이, 임의의 하위 비트 평면이 존재하면 하나 이상의 코드워드들 "acod_r[r]"이 존재하는 것을 야기하는데, 그 각각은 제1 스펙트럼 값의 최하위 비트 평면의 1 비트를 표현하고, 그 각각은 또한 제2 스펙트럼 값의 최하위 비트 평면의 1 비트를 표현한다. 하나 이상의 코드워드들 "acod_r[r]"은, 예를 들어, 상수이며 콘텍스트에 독립적일(context independent) 수 있는 상응하는 누적 빈도 테이블에 따라 인코딩된다. 그러나, 하나 이상의 코드워드들 "acod_r[r]"의 디코딩하기 위한 누적 빈도 테이블의 선택을 위해 각각 다른 매커니즘들이 가능하다.

[0395] 또한, 도면 부호 668에서 보여진 바와 같이, 스펙트럼 값들의 각각의 튜플의 인코딩 이후에 콘텍스트가 업데이트되어, 스펙트럼 값들의 2 개의 이어지는 튜플들을 인코딩 및 디코딩하기 위한 콘텍스트는 일반적으로 다르다는 것을 알아야 한다.

[0396] 도 6i는 정의들에 대한 범례 및 산술적으로 인코딩된 데이터 블록의 구문을 정의하는 조력 성분들을 도시한다.

[0397] 또한, 도 6j에 도시된 상응하는 정의들에 대한 범례 및 조력 성분들과 함께, 산술 데이터 "arith_data()"의 대

안적인 구문이 도 6h에 도시된다.

[0398] 상기를 요약하면, 오디오 인코더(100)에 의해 제공될 수 있고 오디오 디코더(200)에 의해 평가될 수 있는 비트스트림 형식이 기술되었다. 산술적으로 인코딩된 스펙트럼 값들의 비트스트림은 상기에서 논의된 디코딩 알고리즘에 적합하도록 인코딩된다.

[0399] 또한, 인코딩은 디코딩의 역 연산이어서, 인코더가 상기에서 논의된 테이블들을 이용하여 테이블 검색(lookup)을 수행하는데, 이는 디코더에 의해 수행된 테이블 검색의 거의 역인 것으로 일반적으로 여겨질 수 있음을 일반적으로 알아야 한다. 일반적으로, 디코딩 알고리즘 및/또는 바라는 비트스트림 구문을 아는 당업자들은 비트스트림 구문에서 정의되고 산술 디코더에 의해 요구되는 데이터를 제공하는 산술 인코더를 쉽게 설계할 수 있다고 할 수 있다.

[0400] 또한, 수치적 현재 콘텍스트 값을 결정하고 맵핑 규칙 인덱스 값을 도출하기 위한 매커니즘들은 오디오 인코더와 오디오 디코더에서 동일할 수 있음을 알아야 하는데, 이는 일반적으로 오디오 디코더가 오디오 인코더와 동일한 콘텍스트를 이용하는 것이 요구되기 때문으로, 디코딩이 인코딩에 적응된다.

[0401] 15. 구현 대안들

[0402] 비록 몇몇 양상들이 장치의 맥락에서 기술되었으나, 이러한 양상들은 또한 상응하는 방법에 대한 설명을 나타내는 것이 자명한데, 블록 또는 장치는 방법 단계 또는 방법 단계의 특징에 상응한다. 비슷하게, 방법 단계의 맥락에서 기술된 양상들은 또한 상응하는 블록 또는 항목 또는 상응하는 장치의 특징에 대한 설명을 나타낸다. 몇몇 또는 모든 방법 단계들은, 예를 들어, 마이크로프로세서, 프로그램 가능한 컴퓨터, 또는 전자 회로와 같은 하드웨어 장치로(또는 하드웨어 장치를 이용하여) 실행될 수 있다. 몇몇 실시예들에서, 어떤 하나 이상의 가장 중요한 방법 단계들이 그러한 장치에 의해 실행될 수 있다.

[0403] 본 발명의 인코딩된 오디오 신호는 디지털 저장 매체에 저장될 수 있거나, 인터넷과 같은 무선 전송 매체 또는 유선 전송 매체로 전송될 수 있다.

[0404] 특정 구현 요구조건들에 따라, 본 발명의 실시예들은 하드웨어 또는 소프트웨어로 구현될 수 있다. 상기 구현은, 각각의 방법이 수행되도록 프로그램 가능한 컴퓨터 시스템과 협력하는(또는 협력할 수 있는), 전자적으로 판독가능한 제어 신호들이 그 위에 저장된 디지털 저장 매체, 예를 들어, 플로피 디스크, DVD, 블루레이, CD, ROM, PROM, EPROM, EEPROM, 또는 플레쉬 메모리를 이용하여 수행될 수 있다. 그러므로, 디지털 저장 매체는 컴퓨터 판독가능할 수 있다.

[0405] 본 발명에 따른 몇몇 실시예들은 프로그램 가능한 컴퓨터 시스템과 협력 가능한 전자적으로 판독가능한 제어 신호들을 갖는 데이터 캐리어(carrier)를 포함하여, 여기서 기술된 방법들 중 하나가 수행된다.

[0406] 일반적으로, 본 발명의 실시예들은 프로그램 코드를 갖는 컴퓨터 프로그램 제품으로 구현될 수 있는데, 컴퓨터 프로그램 제품이 컴퓨터에서 구동할 때 프로그램 코드는 상기 방법들 중 하나를 수행하기 위해 작동된다. 프로그램 코드는, 예를 들어, 기계 판독가능한 캐리어에 저장될 수 있다.

[0407] 다른 실시예들은, 기계 판독가능한 캐리어에 저장된, 여기서 기술된 방법들 중 하나를 수행하기 위한 컴퓨터 프

로그램을 포함한다.

[0408] 다시 말해서, 본 발명의 방법에 대한 일 실시예는, 그러므로, 컴퓨터 프로그램이 컴퓨터에서 구동할 때, 여기서 기술된 방법들 중 하나를 수행하기 위한 프로그램 코드를 갖는 컴퓨터 프로그램이다.

[0409] 본 발명의 방법들에 대한 추가적인 실시예는, 그러므로, 여기서 기술된 방법들 중 하나를 수행하기 위한 컴퓨터 프로그램이 그 위에 기록된 것을 포함하는 데이터 캐리어(또는 디지털 저장 매체, 또는 컴퓨터 판독가능한 매체)이다. 데이터 캐리어, 디지털 저장 매채, 또는 기록된 매체는 일반적으로 유형이고/유형이거나 변하지 않는다.

[0410] 본 발명 방법에 대한 추가적인 실시예는, 그러므로, 여기서 기술된 방법들 중 하나를 수행하기 위한 컴퓨터 프로그램을 표현하는 데이터 스트림 또는 신호들의 시퀀스이다. 데이터 스트림 또는 신호들의 스퀀스는, 예를 들어, 데이터 통신 연결, 예를 들어, 인터넷을 통해 전송되도록 구성될 수 있다.

[0411] 추가적인 실시예는 여기서 기술된 방법들 중 하나를 수행하도록 구성되거나 적응된 처리 수단, 예를 들어, 컴퓨터 또는 프로그램 가능한 논리 소자를 포함한다.

[0412] 추가적인 실시예는 여기서 기술된 방법들 중 하나를 수행하기 위한 컴퓨터 프로그램이 그 위에 설치된 컴퓨터를 포함한다.

[0413] 본 발명에 따른 추가적인 실시예들은, 수신기로, 여기서 기술된 방법들 중 하나를 수행하기 위한 컴퓨터 프로그램을 (예를 들어, 전자적으로 또는 광학적으로) 전송하도록 구성된 장치 또는 시스템을 포함한다. 상기 수신기는, 예를 들어, 컴퓨터, 이동 기기, 메모리 소자 등등 일 수 있다. 상기 장치 또는 시스템은, 예를 들어, 수신기로 컴퓨터 프로그램에 전송하기 위한 파일 서버를 포함할 수 있다.

[0414] 몇몇 실시예들에서, 프로그램 가능한 논리 소자(예를 들어, 필드 프로그램 게이트 어레이)는 여기서 기술된 방법의 몇몇 또는 모든 기능들을 수행하는데 이용될 수 있다. 몇몇 실시예들에서, 필드 프로그램 가능한 게이트 어레이는 여기서 기술된 방법들 중 하나를 수행하기 위해 마이크로프로세서와 협력할 수 있다. 일반적으로, 상기 방법들은 바람직하게는 임의의 하드웨어 장치로 수행된다.

[0415] 상기에서 기술된 실시예들은 단지 본 발명의 원리들에 대한 예시일 뿐이다. 여기서 기술된 배치 및 세부사항들에 대한 수정 및 변경이 당업자에게 자명할 것으로 이해된다. 그러므로, 오직 곧 있을 특허 청구항들의 범위에 의해서만 제한되고, 여기에서의 실시예들에 대한 기술 및 설명으로 표현된 특정 세부사항들에 의해 제한되지 않음을 의도한다.

[0416] 16. 결론

[0417] 결론적으로 말하면, 본 발명에 따른 실시예들은 다음의 양상들을 하나 이상 포함하는데, 여기서 상기 양상들은 개별적으로 또는 결합하여 이용될 수 있다.

[0418] a) 콘텍스트 상태 해싱 매커니즘

[0419] 본 발명의 일 양상에 따라, 해시 테이블에 상태들이 유효 상태들 및 그룹 경계들로 여겨진다. 이는 요구되는 테이블들의 크기를 상당히 감소시키는 것을 가능하게 한다.

[0420] b) 증분(incremental) 콘텍스트 업데이트

[0421] 일 양상에 따라, 본 발명에 따른 몇몇 실시예들은 콘텍스트를 업데이트하기 위한 계산 효율적인 방식을 포함한다. 몇몇 실시예들은 수치적 현재 콘텍스트 값이 수치적 이전 콘텍스트 값으로부터 도출되는 증분 콘텍스트 업데이트를 이용한다.

[0422] c) 콘텍스트 도출

[0423] 본 발명의 일 양상에 따라, 2 개의 스펙트럼 절대 값들의 합을 이용하는 것은 단절을 연결시키는 것이다. 이는 (종래의 형상 이득 벡터 양자화와 반대로) 일종의 스펙트럼 계수들의 이득 벡터 양자화이다. 이는 콘텍스트 순서를 제한하는 한편, 근처로부터의 가장 의미 있는 정보를 전달하는 것을 목표로 한다.

[0424] 본 발명에 따른 실시예들에 적용되는 몇몇 다른 기술들이 선공개 되지 않은 특허 출원 PCT/EP2010/065725, PCT/EP2010/065726, 및 PCT/EP2010/065727에 기술된다. 또한, 본 발명에 따른 몇몇 실시예들에서, 중지 심볼이 이용된다. 또한, 몇몇 실시예들에서, 오직 무부호 값들만이 콘텍스트를 위해 고려된다.

[0425] 그러나, 상기에서 언급한 선공개되지 않은 국제 특허 출원들은 본 발명에 따른 몇몇 실시예들에서 여전히 이용하고 있는 양상들을 개시한다.

[0426] 예를 들어, 본 발명의 몇몇 실시예들에서 0 구역의 식별이 이용된다. 이에 따라, 이른바 "작은 값 플래그"가 설정된다(예를 들어, 수치적 현재 콘텍스트 값 c의 비트 16).

[0427] 몇몇 실시예들에서, 구역에 따르는(region-dependent) 콘텍스트 계산이 이용될 수 있다. 그러나, 다른 실시예들에서, 복잡도 및 테이블들의 크기를 상당히 작게 유지하기 위해 구역에 따르는 콘텍스트 계산은 생략될 수 있다.

[0428] 또한, 해시 함수를 이용하는 콘텍스트 해싱은 본 발명의 중요한 양상이다. 콘텍스트 해싱은 상기에서 참조된 선공개되지 않은 국제 특허 출원들에서 기술되는 2 개의 테이블 구상에 기초할 수 있다. 그러나, 계산 효율성을 증가시키기 위해 몇몇 실시예들에서 콘텍스트 해싱에 대한 특정 적응이 이용될 수 있다. 그럼에도 불구하고, 본 발명에 따른 몇몇 다른 실시예들에서, 상기에서 참조된 선공개되지 않은 국제 특허 출원들에서 기술되는 콘텍스트 해싱이 이용될 수 있다.

[0429] 또한, 증분 콘텍스트 해싱이 오히려 간단하고 계산 효율적임을 알아야 한다. 또한, 본 발명의 몇몇 실시예들에서 이용되는, 값들의 부호로부터의 콘텍스트 독립은 콘텍스트를 간소화하도록 도움으로써, 메모리 요구를 상당히 낮게 유지한다.

[0430] 본 발명의 몇몇 실시예들에서는, 2 개의 스펙트럼 값들의 합을 이용하는 콘텍스트 도출 및 콘텍스트 제한이 이용된다. 이러한 2 가지의 양상들은 결합될 수 있다. 둘 다 근처로부터 가장 의미있는 정보를 나름으로써 콘텍스트 순서를 제한하는 것을 목표로 한다.

[0431] 몇몇 실시예들에서, 복수의 0 값들의 그룹에 대한 식별과 유사할 수 있는 small-value-flag가 이용된다.

[0432] 본 발명에 따른 몇몇 실시예들에서, 산술 중지 매커니즘이 이용된다. 상기 구상은, 비교가능한 함수를 갖는 JPEG에서의 심볼 "end-of-block"의 사용과 유사하다. 그러나, 본 발명의 몇몇 실시예들에서, 상기 심볼("ARITH_STOP")은 엔트로피 코더에 명백히 포함되지는 않는다. 대신에, 이전에 발생할 수 없는, 이미 존재하는 심볼들의 결합, 즉, "ESC+0"이 이용된다. 다시 말해서, 오디오 디코더는, 수치적 값을 표현하기 위해 일반적으로 이용되지 않는, 존재하는 심볼들의 결합을 감지하고, 산술 중지 조건으로써 이미 존재하는 심볼들의 그런 결합의 발생을 해석하도록 구성된다.

[0433] 본 발명에 따른 일 실시예는 2 개의 테이블 콘텍스트 해싱 매커니즘을 이용한다.

[0434] 더 요약하면, 본 발명에 따른 몇몇 실시예들은 다음의 4 가지 주요 양상들 중 하나 이상을 포함할 수 있다.

[0435] ● 근처에서 0 구역들 또는 작은 진폭 구역들을 감지하기 위한 확장된 콘텍스트;

[0436] ● 콘텍스트 해싱;

[0437] ● 콘텍스트 상태 생성: 콘텍스트 상태의 증분 업데이트; 및

[0438] ● 콘텍스트 도출: 진폭과 제한의 합을 포함하는 콘텍스트 값들에 대한 특정 양자화.

[0439] 추가로 결론을 말하자면, 본 발명에 따른 실시예들의 일 양상은 증분 콘텍스트 업데이트에 있다. 본 발명에 따른 실시예들은, 규격 초안(예를 들어, 규격 초안 5)의 막대한 계산들을 피하는, 콘텍스트의 업데이트에 대한 효율적인 구상을 포함한다. 오히려, 간단한 이동 연산들 및 논리 연산들이 몇몇 실시예들에서 이용된다. 간단한 콘텍스트 업데이트는 콘텍스트의 계산을 상당히 용이하게 한다.

[0440] 몇몇 실시예들에서, 콘텍스트는 값들(예를 들어, 디코딩된 스펙트럼 값들)의 부호로부터 독립된다. 값들의 부호로부터의 콘텍스트의 이러한 독립은 콘텍스트 변수의 복잡도가 감소되게 한다. 이러한 구상은 콘텍스트에서의 부호의 무시가 코딩 효율성의 심각한 저하를 가져오지는 않는다는 결과에 기초한다.

[0441] 본 발명의 일 양상에 따라, 두 개의 스펙트럼 값들의 합을 이용하여 콘텍스트가 도출된다. 이에 따라, 콘텍스트의 저장을 위한 메모리 요구가 상당히 감소된다. 이에 따라, 두 개의 스펙트럼 값들의 합을 표현하는 콘텍스트 값의 사용이 몇몇 경우에서 유리하게 여겨질 수 있다.

[0442] 또한, 콘텍스트 제한은 몇몇 경우에서 상당한 개선을 가져온다. 두 개의 스펙트럼 값들의 합을 이용하는 콘텍스트의 도출에 더해, 콘텍스트 어레이 "q"의 엔트리들은 몇몇 실시예에서 최대 값 "0xF"으로 제한되는데, 이는 결국 메모리 요구의 제한을 야기한다. 콘텍스트 어레이 "q"의 값들에 대한 이러한 제한은 몇몇 이점들을 가져온다.

[0443] 몇몇 실시예들에서, 이른바 "small value flag"가 이용된다. (수치적 현재 콘텍스트 값이라고도 지칭되는) 콘텍

스트 변수 c의 획득에서, 만약 어떤 엔트리들 "q[1][i-3]" 내지 "q[1][i-1]"의 값들이 매우 작으면, 플래그가 설정된다. 이에 따라, 콘텍스트의 계산이 높은 효율성으로 수행될 수 있다. 특히 의미 있는 콘텍스트 값(예를 들어, 수치적 현재 콘텍스트 값)이 획득될 수 있다.

[0444] 몇몇 실시예들에서, 산술 중지 매커니즘이 이용된다. "ARITH_STOP" 매커니즘은, 만약 오직 0 값들만 남아 있으면, 산술 인코딩 또는 디코딩의 효율적인 중지를 가능하게 한다. 이에 따라, 복잡도의 면에서 보통의 비용으로 코딩 효율성이 개선될 수 있다.

[0445] 본 발명의 일 양상에 따라, 두 개의 테이블 콘텍스트 해싱 매커니즘이 이용된다. 콘텍스트의 맵핑은 테이블 "ari_lookup_m"의 이어지는 검색 테이블 평가와 결합하여 테이블 "ari_hash_m"을 평가하는 구간 분할 알고리즘을 이용하여 수행된다. 이 알고리즘은 WD3 알고리즘보다 더 효율적이다.

[0446] 다음에서는, 몇몇 추가적인 세부사항들이 논의될 것이다.

[0447] 여기서 테이블들 "arith_hash_m[600]" 및 "arith_lookup_m[600]"는 두 개의 구별되는 테이블들임을 알아야 한다. 첫 번째 것은 확률 모델 인덱스(예를 들어, 맵핑 규칙 인덱스 값)에, 단일 콘텍스트 인덱스(예를 들어, 수치적 콘텍스트 값)를 맵핑하는데 이용되고, 두 번째 것은 단일 확률 모델에, "arith_hash_m[]"에서 콘텍스트 인덱스들에 의해 한계가 정해진 연이은 콘텍스트들의 그룹을 맵핑하는데 이용된다.

[0448] 크기가 약간 다를지라도, 테이블 "arith_cf_msb[96][16]"이 테이블 "ari_cf_m[96][17]"의 대안으로 이용될 수 있음을 추가로 알아야 한다.

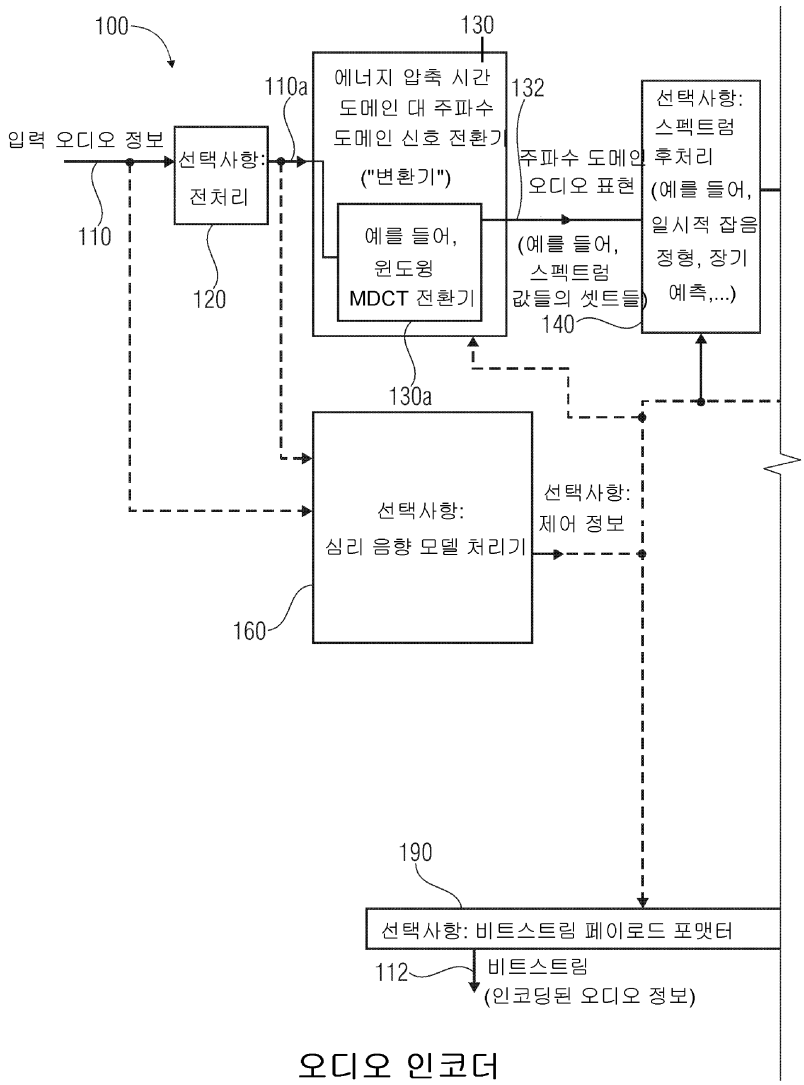
[0449] 확률 모델들의 17 번째 계수들이 항상 0이므로, "ari_cf_m[][]" 및 "ari_cf_msb[][]"는 동일한 테이블을 참조할 수 있다. 이는 테이블들의 저장을 위해 요구되는 공간을 계산할 때 때때로 계산되지 않는다.

[0450] 상기를 요약하면, 본 발명에 따른 몇몇 실시예들은, MPEG USAC 규격 초안(예를 들어, MPEG USAC 규격 초안 5)에 수정을 가한 제안된 새로운 무잡음 코딩(인코딩 또는 디코딩)을 제공한다. 상기 수정은 첨부된 도면들 및 또한 관련 설명에서 알 수 있다.

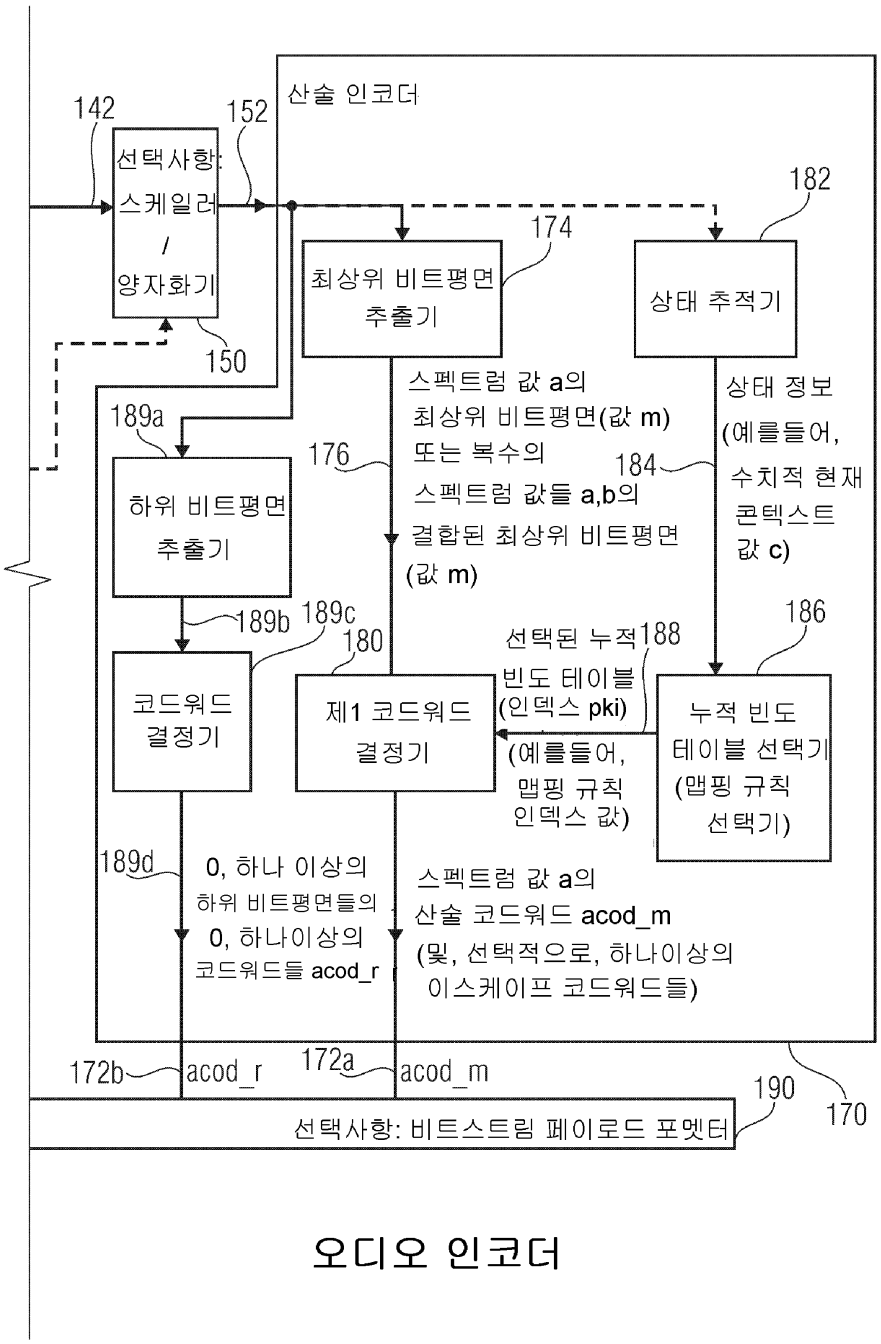
[0451] 끝맺는 말로써, 변수들, 어레이들, 함수들, 기타 등등의 명칭들에서 접두사 "ari" 및 접두사 "arith"는 교체가능하게 이용된다는 것을 알아야 한다.

도면

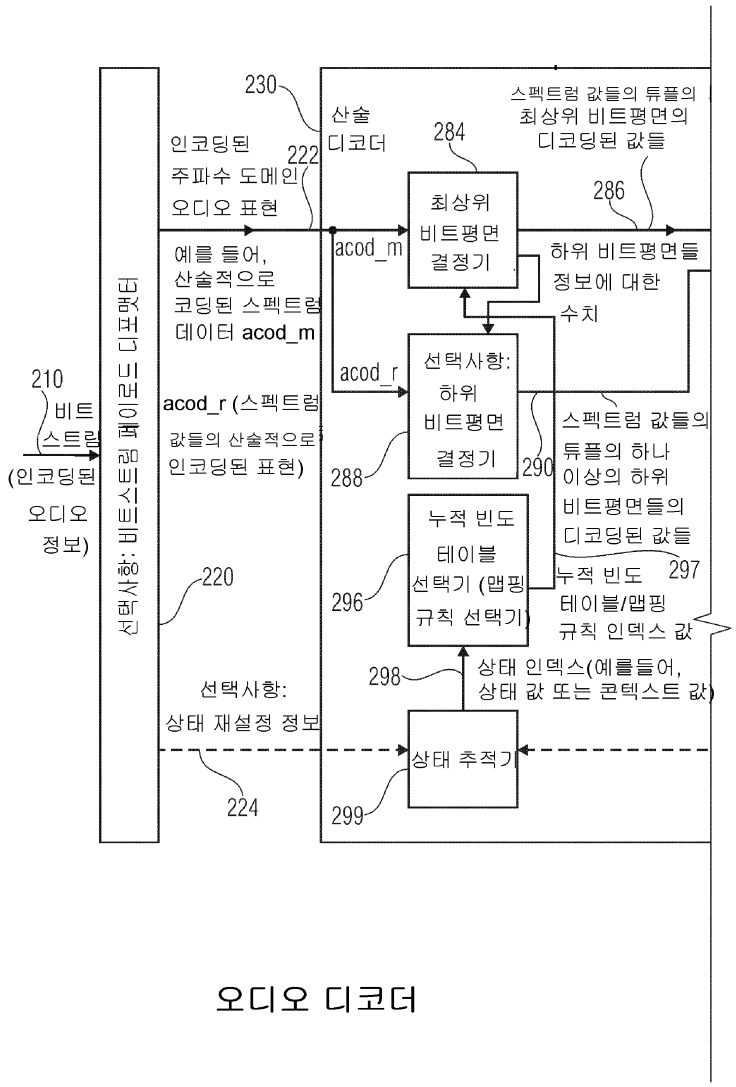
도면1a


100
130
110a
132
에너지 압축 시간
도메인 대 주파수
도메인 신호 전환기
("변환기")
선택사항:
스펙트럼
후처리
(예를 들어,
일시적 잡음
정형, 장기
예측,...)
입력 오디오 정보
선택사항:
전처리
주파수 도메인
오디오 표현
예를 들어,
윈도윙
MDCT 전환기
(예를 들어,
스펙트럼
값들의 셋트들)
110
120
140
130a
선택사항:
심리 음향 모델 처리기
선택사항:
제어 정보
160
190
선택사항: 비트스트림 페이로드 포맷터
112
비트스트림
(인코딩된 오디오 정보)
오디오 인코더

도면1b

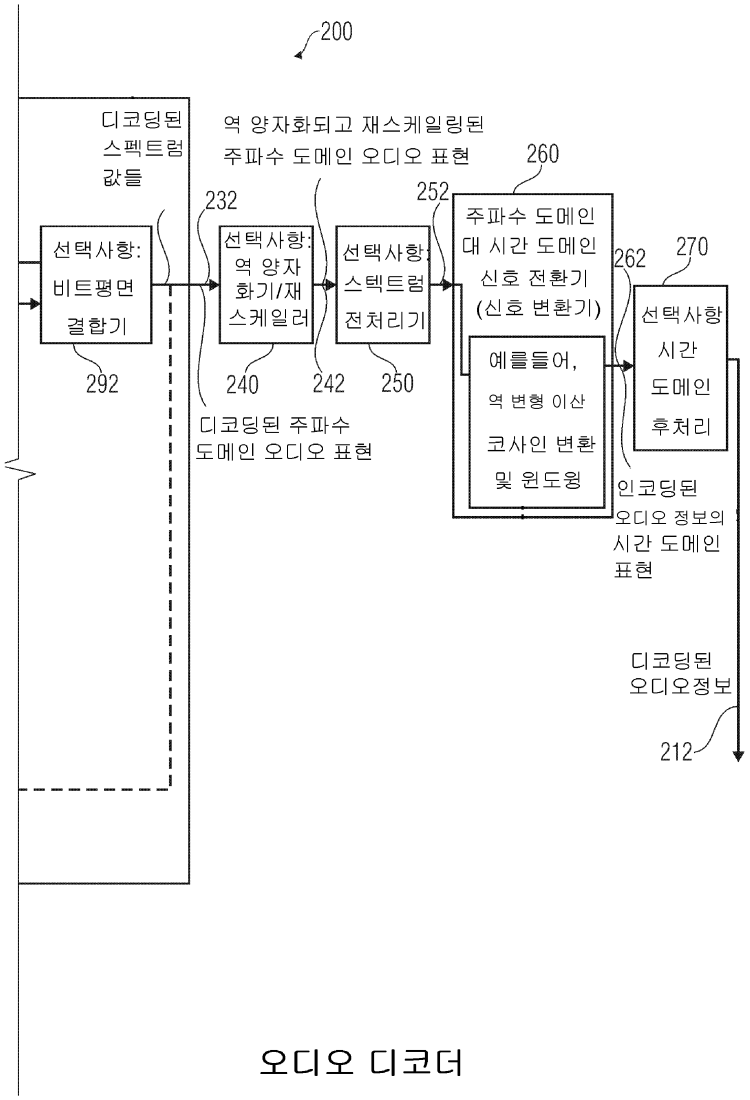

산술 인코더
142
152
선택사항:
스케일러
/
양자화기
150
182
174
최상위 비트평면
추출기
상태 추적기
스펙트럼 값 a의
최상위 비트평면(값 m)
또는 복수의
스펙트럼 값들 a,b의
결합된 최상위 비트평면
(값 m)
상태 정보
(예를들어,
수치적 현재
콘텍스트
값 c)
189a
176
184
하위 비트평면
추출기
189b
189c
180
188
186
선택된 누적
빈도 테이블
(인덱스 pki)
코드워드
결정기
제1 코드워드
결정기
누적 빈도
테이블 선택기
(맵핑 규칙
선택기)
(예를들어,
맵핑 규칙
인덱스 값)
189d
0, 하나 이상의
하위 비트평면들의
0, 하나이상의
코드워드들 acod_r
스펙트럼 값 a의
산술 코드워드 acod_m
(및, 선택적으로, 하나이상의
이스케이프 코드워드들)
172b
acod_r
172a
acod_m
190
170
선택사항: 비트스트림 페이로드 포멧터
오디오 인코더

도면2a


230
산술
디코더
284
스펙트럼 값들의 튜플의
최상위 비트평면의
디코딩된 값들
인코딩된
주파수 도메인
오디오 표현
222
최상위
비트평면
결정기
286
acod_m
하위 비트평면들
정보에 대한
수치
예를 들어,
산술적으로
코딩된 스펙트럼
데이터 acod_m
acod_r
선택사항:
하위
비트평면
결정기
210
비트
스트림
(인코딩된
오디오
정보)
acod_r (스펙트럼
값들의 산술적으로
인코딩된 표현)
288
290
스펙트럼 값들의
튜플의 하나
이상의 하위
비트평면들의
디코딩된 값들
누적 빈도
테이블
선택기 (맵핑
규칙 선택기)
296
누적 빈도
테이블/맵핑
규칙 인덱스 값
297
220
298
상태 인덱스(예를들어,
상태 값 또는 콘텍스트 값)
선택사항:
상태 재설정 정보
상태 추적기
224
299
선택사항: 비트스트림 페이로드 디포맷터

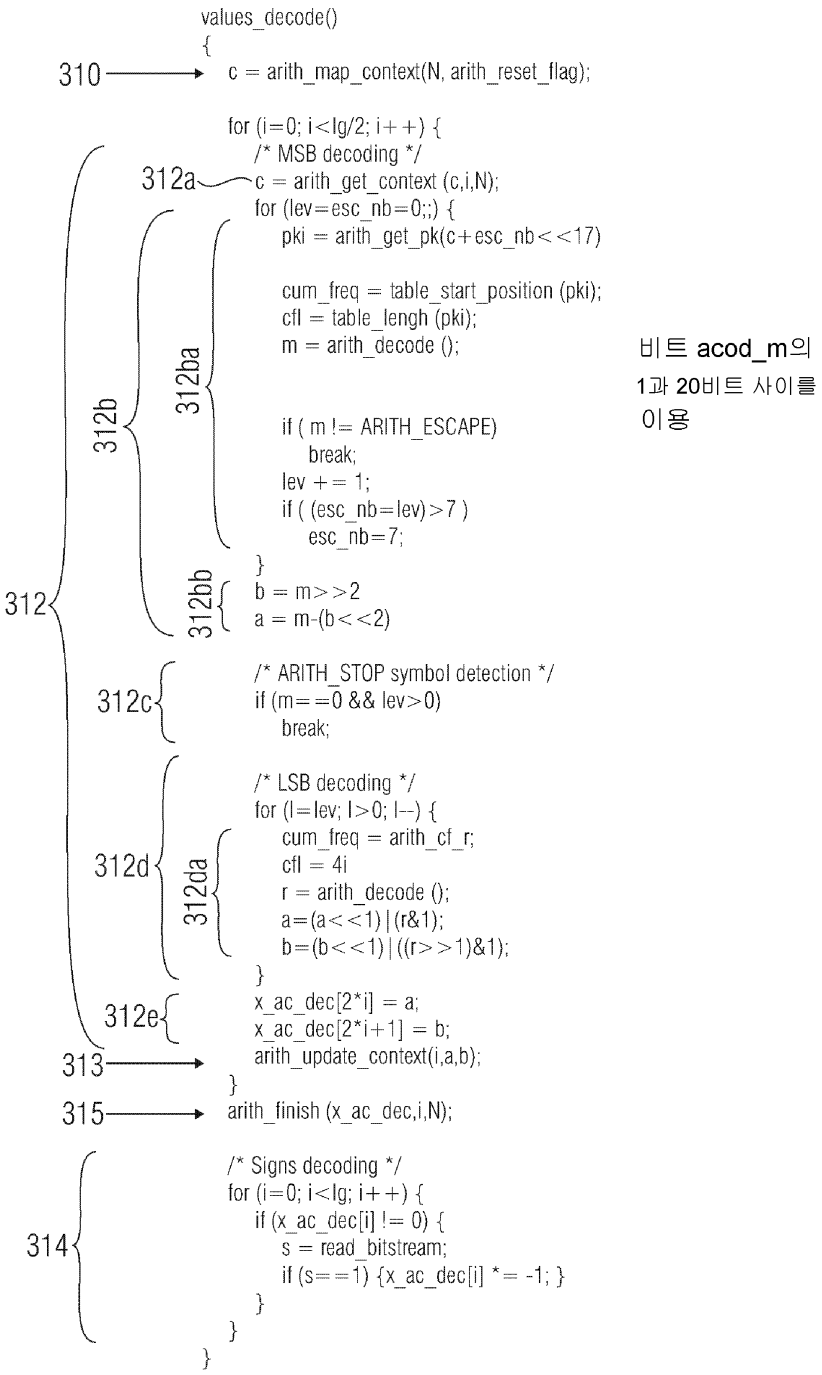
오디오 디코더

도면2b

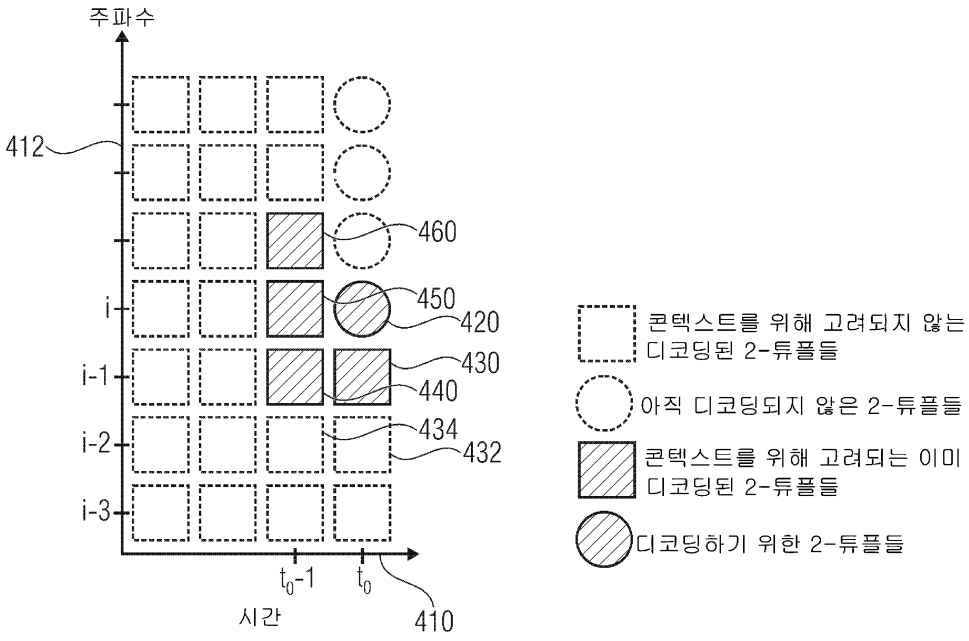

200
디코딩된
스펙트럼
값들
선택사항:
비트평면
결합기
292
232
역 양자화되고 재스케일링된
주파수 도메인 오디오 표현
선택사항:
역 양자
화기/재
스케일러
240
242
252
선택사항:
스펙트럼
전처리기
250
260
주파수 도메인
대 시간 도메인
신호 전환기
(신호 변환기)
예를들어,
역 변형 이산
코사인 변환
및 윈도윙
262
270
선택사항
시간
도메인
후처리
디코딩된 주파수
도메인 오디오 표현
인코딩된
오디오 정보의
시간 도메인
표현
디코딩된
오디오정보
212

오디오 디코더

도면3


values_decode()
{
310
c = arith_map_context(N, arith_reset_flag);
for (i=0; i<lg/2; i++) {
/* MSB decoding */
312a
c = arith_get_context (c,i,N);
for (lev=esc_nb=0;;) {
312b
312ba
pki = arith_get_pk(c+esc_nb<<17)
cum_freq = table_start_position (pki);
cfl = table_lengh (pki);
m = arith_decode ();
비트 acod_m의 1과 20비트 사이를 이용
if (m != ARITH_ESCAPE)
break;
lev += 1;
if ((esc_nb=lev)>7)
esc_nb=7;
}
312
312bb
b = m>>2
a = m-(b<<2)
/* ARITH_STOP symbol detection */
312c
if (m==0 && lev>0)
break;
/* LSB decoding */
312d
for (l=lev; l>0; l--) {
312da
cum_freq = arith_cf_r;
cfl = 4i
r = arith_decode ();
a=(a<<1)|(r&1);
b=(b<<1)|((r>>1)&1);
}
312e
x_ac_dec[2*i] = a;
x_ac_dec[2*i+1] = b;
313
arith_update_context(i,a,b);
}
315
arith_finish (x_ac_dec,i,N);
/* Signs decoding */
314
for (i=0; i<lg; i++) {
if (x_ac_dec[i] != 0) {
s = read_bitstream;
if (s==1) {x_ac_dec[i] *= -1; }
}
}
}

도면4



상태 계산을 위한 콘텍스트

도면5a

```
/*Input variables*/
N /*Length of the current window*/
arith_reset_flag /*Arithmetic coder reset flag*/

/*Global variables*/
previous_N /*Length of the previous window */

c = arith_map_context(N,arith_reset_flag)
{
    if (arith_reset_flag) {
        for (j=0; j<N/4; j++) {
            q[0][j]=0;
        }
    } else {
        ratio = ((float)previous_N) / ((float)N);
        for (j=0; j<N/4; j++) {
            k = (int) ((float) j * ratio);
            q[0][j] = q[1][k];
        }
    }

    previous_N=N;

    return(q[0][0]<<12);
}
```

500a

500b

도면5b

```
/*Input variables*/
lg /*Number of sepctral coefficients to decode in the frame*/
arith_reset_flag /*Arithmetic coder reset flag*/
/*Global variables*/
previous_lg /*Previous number of spectral lines of the previous frame*/

c=arith_map_context (lg,arith_reset_flag)
{
   v=w=0

   if(arith_reset_flag){
      for(j=0; j<lg/2; j++){
         q[0][v++]=0;
      }
   else{
      ratio= ((float)previous_lg)/((float)lg);
      for(j=0; j<lg/2; j++){
         k = (int) ((float)) ((j)*ratio);
         q[0][v++] = qs[w+k];
      }
   }

   previous_lg=lg;

   return(q[0][0]<<12);
}
```

도면5c

```
504 ~

            /*Input variables*/
            c /*old state context*/
            i /*Index of the 2-tuple to decode in the vector*/
            N /*Window Length*/

            /*Output value*/
            c /*updated state context*/

            c = arith_get_context(c,i,N)
            {
504a ~      c = c>>4;
            if(i<N/4-1)
504b ~         c = c + (q[0][i+1]<<12);
504c ~      c = (c&0xFFF0);

            if(i>0)
504d ~         c = c + (q[1][i-1]);

            if (i > 3) {
               if ((q[1][i-3] + q[1][i-2] + q[1][i-1]) < 5)
504e ~            return(c+0x10000);
            }

504f ~      return (c);
         }
```

도면5d

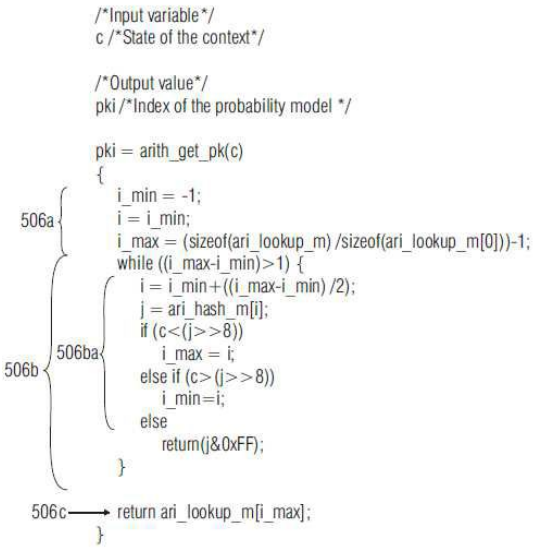
```
/*Input variables*/
c /*old state context*/
i /*Index of the 2-tuple to decode in the vector*/
/*Output value*/
c /*updated state context*/


c=arith_get_context(c,i)
{
    c=c>>4;
    c=(c)+(q[0][i+1]<<12);
    c=(c&0xFFF0)+(q[1][i-1]);

    if(i > 3) {
        if((q[1][i-3] + q[1][i-2] + q[1][i-1]) < 5)
            return(c+0x10000);
    }

    return(c);
}
```

도면5e



도면5f

```
/*Input variable*/
c /*State of the context*/
/*Output value*/
pki /*Index of the probability model */
/*constants*/
i_diff[]={ 299, 149, 74, 37, 18, 9, 4, 2, 1};

pki=arith_get_pk(c) {
    i_min=0;                          508a
    s=c<<8;
    for(k=0;k<9;k++) {                508b
        i=i_min+i_diff[k];            508ba
        j=ari_hash_m[i];
        if(s>j) {
            i_min=i+1;
        }
    }

    j=ari_hash_m[i_min];              508c
    if(s>j)
        return(ari_lookup_m[i_min+1]);
    else if(c<(j>>8))
        return(ari_lookup_m[i_min]);
    else
        return(j&0xFF);
}
```

도면5g

(1)

```
/*helper functions*/
bool arith_first_symbol(void);
    /* Return TRUE if it is the first symbol of the sequence,
       FALSE otherwise */
Ushort arith_gei_next_bit(void);
    /* Get the next bit of the bitstream */

/* global variables */
low
high
value

/* input variables */
cum_freq[]; /* cumulative frequencies table */
cfl; /* length of cum_freq[] */

symbol = arith_decode(cum_freq, cfl)
{
    if (arith_first_symbol()) {           570a
        value = 0;
        for (i=1; i<=16; i++) {
            value = (val<<1) | arith_get_next_bit();
        }
        low  = 0;
        high = 65535;
    }

    range = high-low+1;                   570b
    cum = ((((int) (value-low+1))<<14)-((int) 1))/range;
    p = cum_freq-1;
```

(2)

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

    symbol = p-cum_freq+1;                570d
    if (symbol)                           570e
        high = low - (range*cum_freq[symbol-1])>>14 - 1;

    low += (range * cum_freq[symbol])>>14;

    for (;;) {                            570f
        if (high<32768) {}                570fa
        else if (low>=32768) {
            value -= 32768;
            low  -= 32768;
            high -= 32768;
        }
        else if (low>=16384 && high<49152) {
            value -= 16384;
            low  -= 16384;
            high -= 16384;
        }
        else break;

        low += low;                       570fb
        high += high+1;
        value = (value<<1) | arith_get_next_bit();
    }
    return symbol;
}
```

도면5h

```
/*helper functions*/
bool arith_first_symbol(void);
    /* Return TRUE if it is the first symbol of the sequence,
    FALSE otherwise */
Ushort arith_get_next_bit(void);
    /* Get the next bit of the bitstream */

/* global variables */
low
high
value

/* input variables */
cum_freq[]; /* cumulative frequencies table */
cfl; /* length of cum_freq[] */

symbol = arith_decode(cum_freq, cfl)
{
    if (arith_first_symbol()) {
        value = 0;
        for (i=1; i<=16; i++) {
            value = (val<<1) | arith_get_next_bit();
        }
        low  = 0;
        high  = 65535;
    }

    range = high-low+1;
    cum =((((int) (value-low+1))<<14)-((int) 1));
    p = cum_freq-1;

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

도 5i에서 계속됨

도면5i

도 5h로부터 계속됨

```
while ( cfl>1 );

symbol = p-cum_freq+1;
if (symbol)
    high = low + (range*cum_freq[symbol-1])>>14 - 1;

low += (range * cum_freq[symbol])>>14;

for (;;) {
    if (high<32768) {}
    else if (low>=32768) {
        value -= 32768;
        low   -= 32768;
        high  -= 32768;
    }
    else if (low>=16384 && high<49152) {
        value -= 16384;
        low   -= 16384;
        high  -= 16384;
    }
    else break;

    low += low;
    high += high+1;
    value = (value<<1) | arith_get_next_bit();
}
return symbol;
}
```

도면5j

```
b = m>>2;
a = m-(b<<2);
for (j=0;j<lev;j++) {
    r = arith_decode(arith_cf_r,4);
    a = (a<<1) | (r&1);
    b = (b<<1) | ((r>>1)&1);
}
```

도면5k

```
x_ac_dec[2*i]   = a
x_ac_dec[2*i+1] = b;
```

도면5l

```
/*input variables*/
a,b /* Decoded unsigned quantized spectralcoefficients of the 2-tuple */
i /* Index of the quantized spectral coefficient to decode */

arith_update_context(i, a, b)
{
    q[1][i] = a+b+1;
    if (q[1][i]>0xF)
        q[1][i] = 0xF;
}
```

도면5m

```
/*input variables*/
offset /*number of decoded 2-tuple */
N /*Window length */
x_ac_dec  /*vector of decoded spectal coefficients*/

arith_finish(x_ac_dec,offset,N)
{
    for(i=offset ;i<N/4;i++) {
        x_ac_dec[2*i] = 0;
        x_ac_dec[2*i+1] = 0;
        q[1][i] = 1;
    }
}
```

도면5n

```
b= m>>2
a = m&0x03;
for(j=0;j<lev;j++){
    r = arith_decode(arith_cf_r,4);
    a = (a<<1) | (r&1);
    b = (b<<1) |((r>>1)&1);

}
```

도면5o

```
/*input variables*/
a,b /*Decoded unsigned quantized spectralcoefficients of the 2-tuple*/
i /*Index of the quantized spectral coefficient to decode*/

arith_update_context (){
    qdec[2*i]=a
    qdec[2*i+1]=b;
    q[1][i]=a+b+1;

    if(q[1][i]>0xF)
        q[1][i]=0xF;
}
```

도면5p

```
/*input variables*/
i /*Index of the quantized spectral coefficient to decode*/
lg /*number of coefficients in the frame*/

arith_save_context(i,lg){

    for(;i<N/4;i++){
        qdec[2*i]=0;
        qdec[2*i+1]=0;
        q[1][i]=1;
    }

    if(core_mode==1){
        ratio= ((float) lg)/((float)1024);
        for(j=0; j<512; j++){
            k = (int) ((float) j*ratio);
            qs[j] = q[1][k];
        }
        previous_lg = 512;
    }
    else{
        for(j=0; j<512; j++){
            qs[j] = q[1][j];
        }
        previous_lg = MIN(1024,lg);
    }
}
```

도면5q

정의

a,b	디코딩하기 위한 2-튜플(디코딩하기 위한 2-튜플의 양자화된 계수)
m	디코딩하기 위한 양자화된 스펙트럼 계수의 최상위 2 비트 방식 평면
r	디코딩하기 위한 양자화된 스펙트럼 계수의 하위 비트 평면들
lev	잔여 비트 평면들의 레벨. 이는 하위 비트 평면들의 수에 상응한다.
arith_hash_m[]	누적 빈도 테이블 인덱스 pki에 콘텍스트 상태들을 맵핑하는 해시 테이블
arith_lookup_m[]	누적 빈도 테이블 인덱스 pki에 콘텍스트 상태들의 그룹을 맵핑하는 검색 테이블
arith_cf_m[pki][17]	최상위 2 비트 방식 평면 m 및 ARITH_ESCAPE 심볼에 대한 누적 빈도의 모델들
arith_cf_r [lsbidx][]	최하위 비트 평면들 심볼 r에 대한 누적 빈도
arith_cf_r []	최하위 비트 평면들 심볼 r에 대한 누적 빈도
q[2][]	이전 및 현재 프레임의 2-튜플 콘텍스트 성분들
x_ace_dec[]	디코딩된 양자화된 스펙트럼 계수들
arith_reset_flag	스펙트럼 무잡음 콘텍스트가 재설정되어야 하는지를 가리키는 플래그
ARITH_STOP	ARITH_ESCAPE 심볼 및 m=0의 연속으로 구성되는 중지 심볼. 이것이 발생할 때, 프레임의 나머지가 0 값으로 디코딩된다.
N	윈도우 길이. FD모드에 대해 window_sequence로부터 추론되고, TCX에 대해 N=2*lg이다.
previous_N	이전 윈도우의 길이

도면5r

정의

a,b	디코딩하기 위한 2-튜플 양자화된 계수
m	디코딩하기 위한 양자화된 스펙트럼 계수의 최상위 2 비트 방식 평면
r	디코딩하기 위한 양자화된 스펙트럼 계수의 최상위 2 비트 방식 평면
lev	잔여 비트 평면들의 레벨. 이는 최상위 2 비트 방식 평면 보다는 하위 비트 평면들의 수에 상응한다.
arith_hash_m[]	누적 빈도 테이블 인덱스 pki에 콘텍스트 상태들을 맵핑하는 해시 테이블
arith_lookup_m[]	누적 빈도 테이블 인덱스 pki에 콘텍스트 상태들의 그룹을 맵핑하는 검색 테이블
arith_cf_m[pki][17]	최상위 2 비트 방식 평면 m 및 ARITH_ESCAPE 심볼에 대한 누적 빈도의 모델들
arith_cf_r []	최하위 비트 평면들 심볼 r에 대한 누적 빈도
previous_lg	**산술 디코더에 의해 이전에 디코딩된 전송된 스펙트럼 계수들의 수**
q[2][]	2-튜플들의 현재 콘텍스트는 현재 프레임을 디코딩하기 위해 이용한다.
qs[]	다음 프레임을 위해 저장된 과거 콘텍스트
qdec[]	디코딩된 양자화된 스펙트럼 계수들
arith_reset_flag	스펙트럼 무잡음 콘텍스트가 재설정되어야 하는지를 가리키는 플래그
ARITH_STOP	ARITH_ESCAPE 심볼 및 m=0의 연속으로 구성되는 중지 심볼. 이것이 발생할 때, 프레임의 나머지가 0 값들으로 디코딩된다.
N	윈도우 길이. AAC에 대해 window_sequence로부터 추론되고 (섹션 6.8.3.1 참조), TCX에 대해 N=2.lg이다.

도면6a

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

도면6b

Single_channel_element()의 구문

Syntax	No. of bits	Mnemonic

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

도면6c

Channel_pair_element()의 구문

Syntax	No. of bits	Mnemonic

channel_pair_element()
{
core_mode0
core_mode1
 	
ics_info(); optional: common ics_info for two channels
 	
if (core_mode0 == 1) {
lpd_channel_stream();
}
else {
fd_channel_stream();
}
 	
if (core_mode1 == 1) {
lpd_channel_stream();
}
else {
fd_channel_stream();
 	
}
}
	1
	1
		uimsbf
		uimsbf

도면6d

Ics_info()의 구문

Syntax	No. of bits	Mnemonic
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**
}		
else {		
max_sfb;	**6**	**uimsbf**
}		
}		

선택사항

도면6e

Fd_channel_stream()의 구문

Syntax	No. of bits	Mnemonic
fd_channel_stream()		
{		
global_gain;	**8**	**uimsbf**
ics_info(); (unless included in channel pair element)		
scale_factor_data ();		
ac_spectral_data ();		
}		

도면6f

Ac_spectral_data()의 구문

Syntax	No. of bits	Mnemonic
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

Syntax	No. of bits	Mnemonic
arith_data(lg, arith_reset_flag)		
{		
c = arith_map_context(N, arith_reset_flag);		
for (i=0; i<lg/2; i++) {		
/* MSB decoding */		
c = arith_get_context (c,i,N);		
for (lev=esc_nb=0;;) {		
662 pki = arith_get_pk(c+esc_nb<<17)		
663 **acod_m**[pki][m]	1..20	vlclbf
if (m != ARITH_ESCAPE)		
break;		
664 lev += 1;		
if ((esc_nb=lev)>7)		
esc_nb=7;		
}		
b = m>>2;		
a = m – (b<<2);		
/* ARITH_STOP symbol detection */		
if (m==0 && lev>0)		
break;		
/* LSB decoding */		
for (l=lev; l>0; l--) {		
acod_r[r]	1..20	vlclbf
a=(a<<1)|(r&1);		
b=(b<<1)|((r>>1)&1);		
}		
x_ac_dec[2*i] = a;		
x_ac_dec[2*i+1] = b;		
668 arith_update_context(i,a,b);		
}		
arith_finish (x_ac_dec,lg,N);		
/* Signs decoding */		
for (i=0; i<lg; i++) {		
if (x_ac_dec[i] != 0) {		
s;	1	uimsbf
if (s==1) {x_ac_dec[i] *= -1; }		
}		
}		
}		

도면6h

테이블 Arith_data()의구문

Syntax	No. of bits	Mnemonic
Arith_data(lg, arith_reset_flag){		
c=arith_map_context(lg, arith_reset_flag);		
for (i=0; i<lg/2; i++) {		
/*MSBs decoding*/		
c = arith_get_context (c,i);		
for (lev=esc_nb=0;;) {		
pki = arith_get_pk(c+esc_nb<<17)		
acod_m[pki][m]	1..20	vlclbf
if (m != ARITH_ESCAPE)		
break;		
lev += 1;		
if((esc_nb=lev)>7)		
esc_nb=7;		
}		
b=m>>2;		
a=m-(b<<2);		
/*ARITH_STOP symbol detection*/		
if(m==0 && lev>0)		
break;		
/*LSBs decoding*/		
for (l=lev; l>0; l--) {		
acod_r[r]	1..20	vlclbf
a=(a<<1)|(r&1);		
b=(b<<1)|((r>>1)&1);		
}		
arith_update_context(a,b,i);		
}		
arith_save_arith (l,lg);		
/*Signs decoding*/		
for (i=0; i<lg/2; i++) {		
if(a!=0){		
s;	1	uimsbf
If(s) a=-a;		
}		
if(b!=0){		
s;	1	uimsbf
if(s) b=-b;		
}		
}		
}		

도면6i

정의

arith_data()	스펙트럼 무잡음 코더 데이터를 디코딩하기 위한 데이터 성분
arith_reset_flag	스펙트럼 무잡음 콘텍스트가 재설정되어야 하는지를 가리키는 플래그
acod _m[pki][m]	2-튜플의 양자화된 스펙트럼 계수들의 최상위 2 비트 방식 평면 m의 디코딩을 위해 필요한 산술 코드워드
acod_r[lsbidx][]	2-튜플의 양자화된 스펙트럼 계수의 잔여 비트 평면들 r의 디코딩을 위해 필요한 산술 코드워드
s	널이 아닌 스펙트럼 양자화된 계수의 코딩된 부호

조력 성분

a,b	양자화된 스펙트럼 계수들에 상응하는 2-튜플
m	디코딩하기 위한 2-튜플의 최상위 2 비트 방식 평면
r	디코딩하기 위한 2-튜플의 최하위 비트 평면들
lg	디코딩하기 위한 양자화된 계수들의 수
N	윈도우 길이. FD 모드에 대해 window_sequence로부터 추론되고, TCX에 대해 N=2*lg이다.
i	프레임 내의 디코딩하기 위한 2-튜플들의 인덱스
pki	m을 디코딩하기 위해 산술 디코더에 의해 이용된 누적 빈도 테이블의 인덱스
arith_get_pk ()	코드워드 acod_m[pki][m]을 디코딩하기 위해 필요한 누적 빈도 테이블의 인덱스 pki를 반환하는 함수
c	콘텍스트의 상태
lsbidx	r을 디코딩하기 위해 산술 코더에 의해 이용된 누적빈도 테이블들에 대한 인덱스
lev	최상위 2비트 방식 평면 이후에 디코딩하기 위한 비트 평면들의 레벨
ARITH_ESCAPE	두 개의 최상위 비트 평면들 이후에 디코딩하기 위한 추가적인 비트 평면들을 가리키는 이스케이프 심볼
esc_nb	현재 2-튜플을 위해 이미 디코딩된 ARITH_ESCAPE 심볼의 수. 상기 값은 7로 경계지어진다.
x_ac_dec[]	디코딩된 스펙트럼 계수들을 가지고 있는 성분
arith_map_context()	현재 프레임을 디코딩하기 위해 필요한 콘텍스트들을 초기화한다.
arith_get_context()	현재 2-튜플 m 심볼들을 디코딩하기 위해 콘텍스트 상태를 계산한다.
arith_update_context()	다음 2-튜플을 위해 콘텍스트를 업데이트한다.
arith_finish ()	무잡음 디코딩을 종료한다.

도면6j

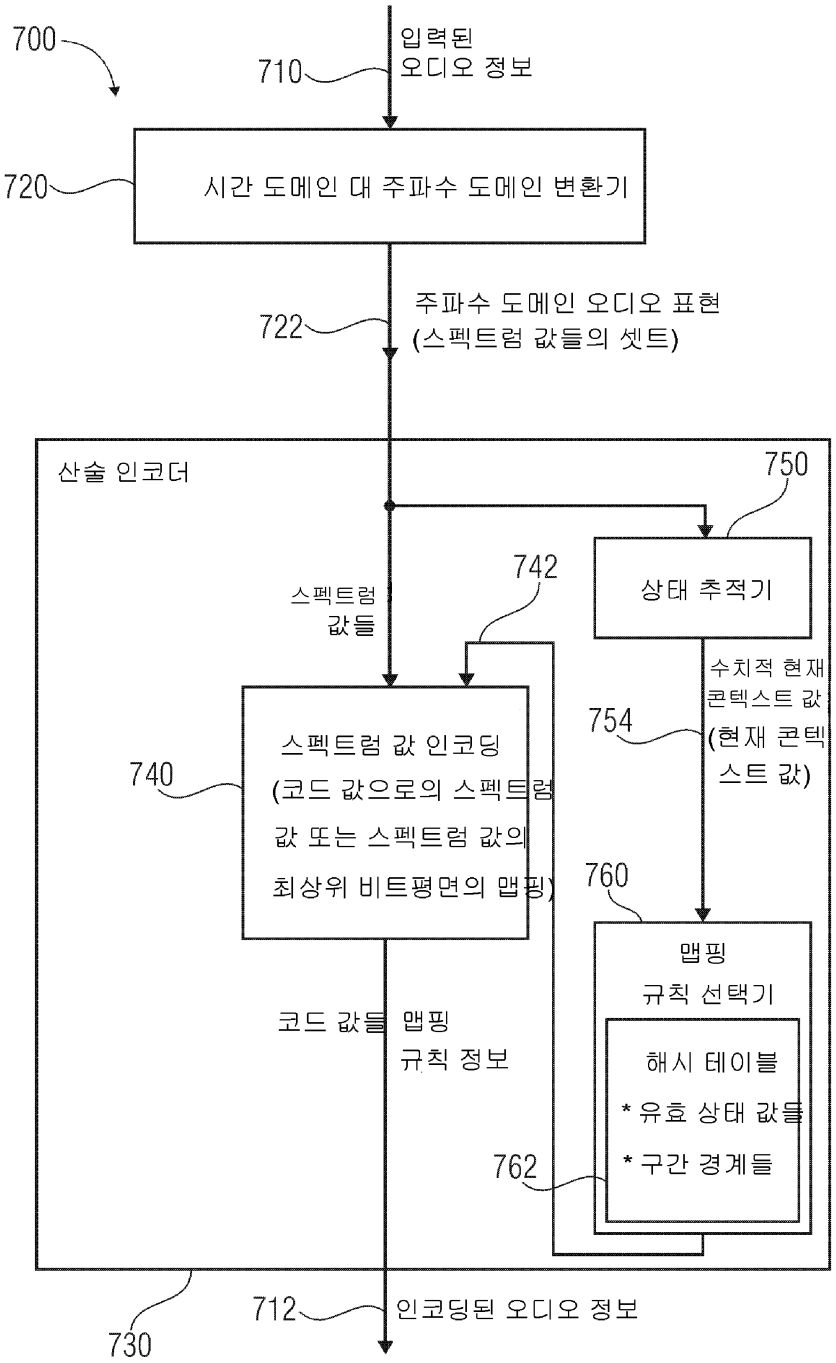
정의

arith_data()	스펙트럼 무잡음 코더 데이터를 디코딩하기 위한 데이터 성분
arith_reset_flag	스펙트럼 무잡음 콘텍스트가 재설정되어야 하는지를 가리키는 플래그
acod _m[pki][m]	2-튜플의 양자화된 스펙트럼 계수들의 최상위 2 비트 방식 평면 m의 디코딩을 위해 필요한 산술 코드워드
arith_r[]	2-튜플의 양자화된 스펙트럼 계수의 잔여 비트 평면들 r의 디코딩을 위해 필요한 산술 코드워드
s	널이 아닌 스펙트럼 양자화된 계수의 코딩된 부호

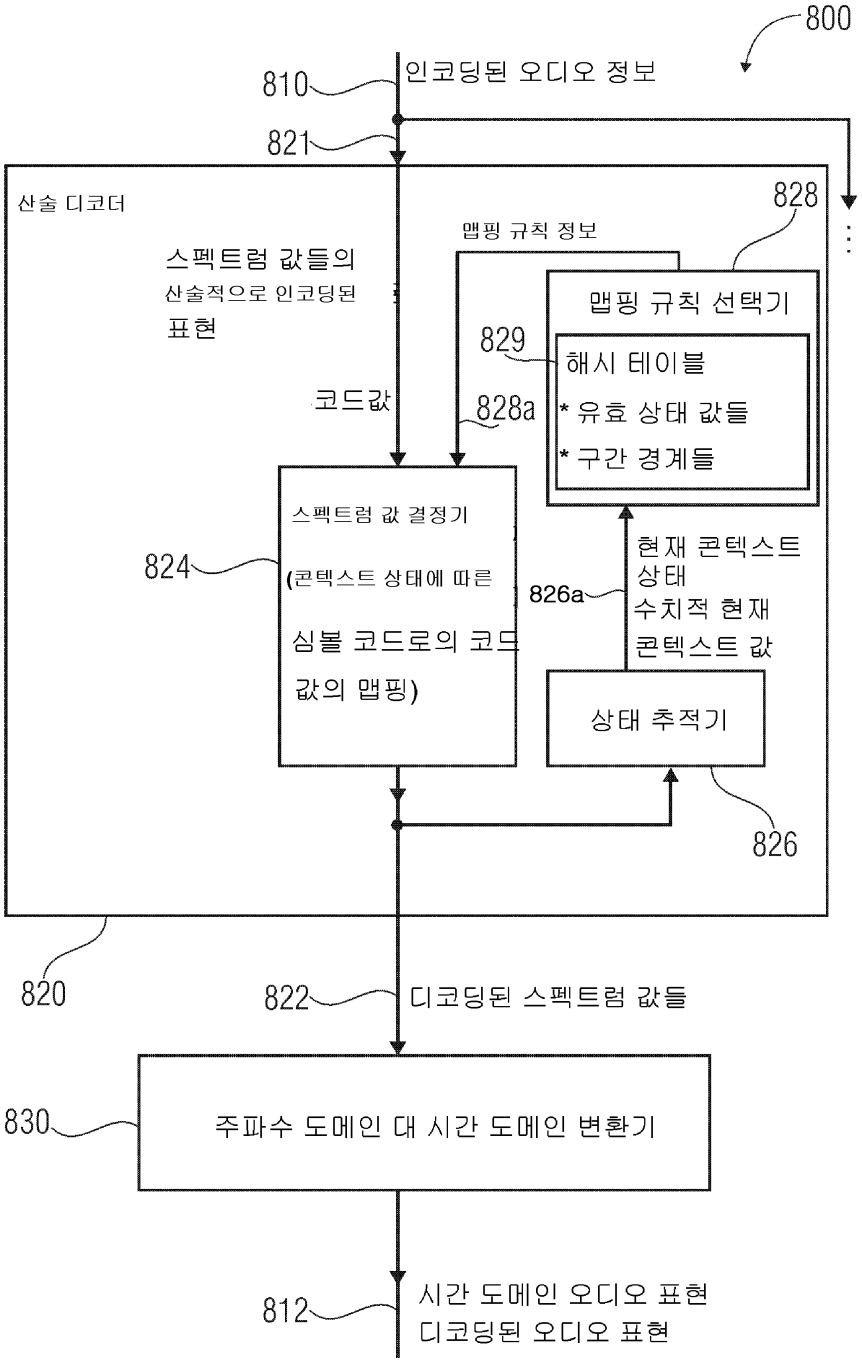
조력 성분

a,b	디코딩하기 위한 2-튜플 양자화된 계수들
m	디코딩하기 위한 2-튜플의 최상위 2 비트 방식 평면
r	디코딩하기 위한 2-튜플의 최하위 비트 방식 평면들
lg	디코딩하기 위한 양자화된 계수들의 수
i	프레임 내의 디코딩하기 위한 2-튜플의 인덱스
pki	m을 디코딩하기 위해 산술 디코더에 의해 이용된 누적 빈도 테이블의 인덱스
arith_get_pk ()	코드워드 acod_m[pki][m]을 디코딩하기 위해 필요한 누적 빈도 테이블의 인덱스 pki를 반환하는 함수
c	콘텍스트의 상태
lev	최상위 2 비트 방식 평면 이후에 디코딩하기 위한 비트 평면들의 레벨
ARITH_ESCAPE	두 개의 최상위 비트 평면들 이후에 디코딩하기 위한 추가적인 비트 평면들을 가리키는 이스케이프 심볼
esc_nb	현재 2-튜플을 위해 이미 디코딩된 ARITH_ESCAPE 심볼의 수 상기 값은 7로 경계지어진다.
arith_map_context()	현재 프레임을 디코딩하기 위해 필요한 콘텍스트들을 초기화한다.
arith_get_context()	현재 2-튜플 m 심볼들을 디코딩하기 위해 콘텍스트 상태를 계산한다
arith_update_context()	다음 2-튜플을 위해 콘텍스트를 업데이트한다
arith_save_context()	디코딩하기 위한 다음 프레임을 위해 콘텍스트를 보관한다.

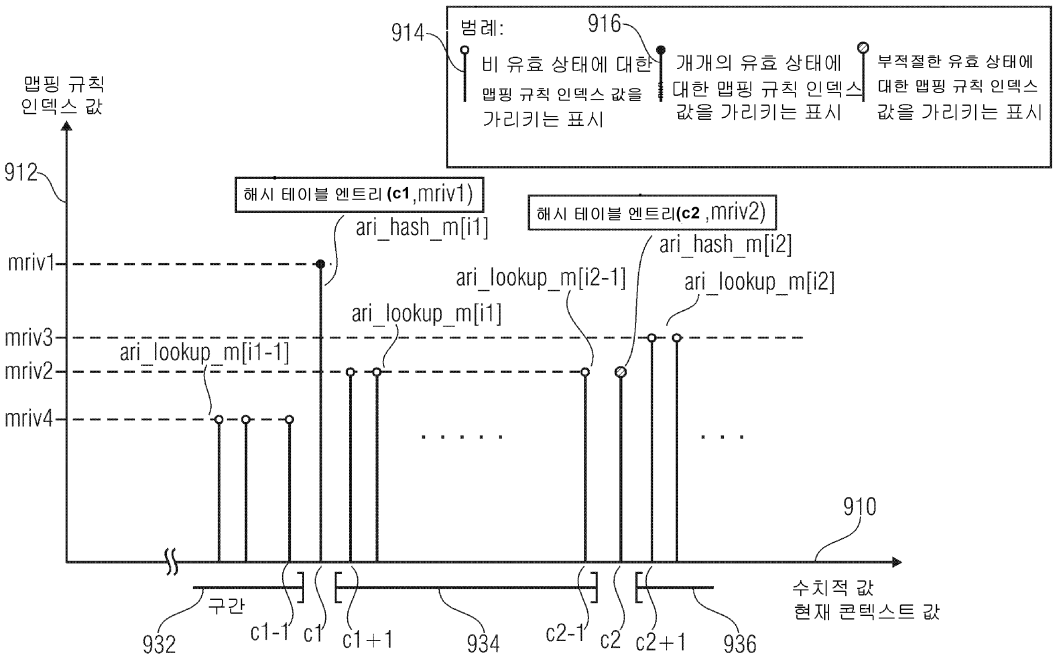
도면7


700
710
입력된
오디오 정보
720
시간 도메인 대 주파수 도메인 변환기
722
주파수 도메인 오디오 표현
(스펙트럼 값들의 셋트)
산술 인코더
750
상태 추적기
742
스펙트럼
값들
수치적 현재
콘텍스트 값
754
(현재 콘텍
스트 값)
740
스펙트럼 값 인코딩
(코드 값으로의 스펙트럼
값 또는 스펙트럼 값의
최상위 비트평면의 맵핑)
760
맵핑
규칙 선택기
해시 테이블
* 유효 상태 값들
* 구간 경계들
762
코드 값들
맵핑
규칙 정보
712
인코딩된 오디오 정보
730

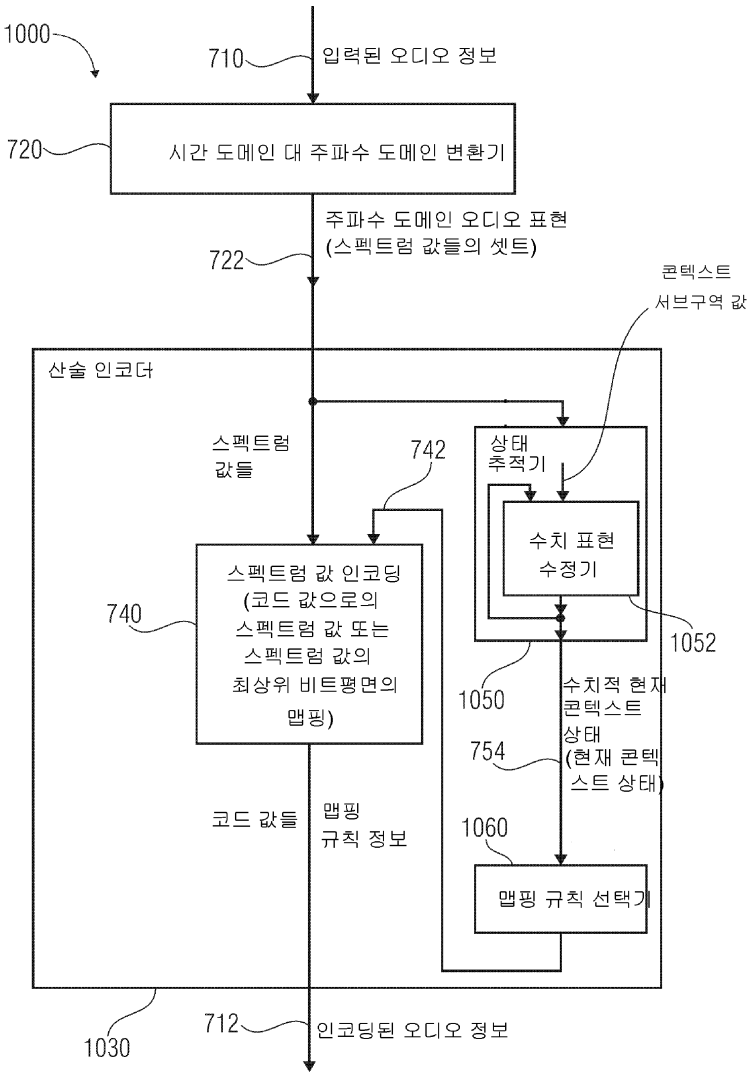
도면8


800
810
인코딩된 오디오 정보
821
산술 디코더
828
맵핑 규칙 정보
스펙트럼 값들의
산술적으로 인코딩된
표현
맵핑 규칙 선택기
829
해시 테이블
* 유효 상태 값들
* 구간 경계들
코드값
828a
스펙트럼 값 결정기
824
(콘텍스트 상태에 따른
심볼 코드로의 코드
값의 맵핑)
826a
현재 콘텍스트
상태
수치적 현재
콘텍스트 값
상태 추적기
826
820
822
디코딩된 스펙트럼 값들
830
주파수 도메인 대 시간 도메인 변환기
812
시간 도메인 오디오 표현
디코딩된 오디오 표현

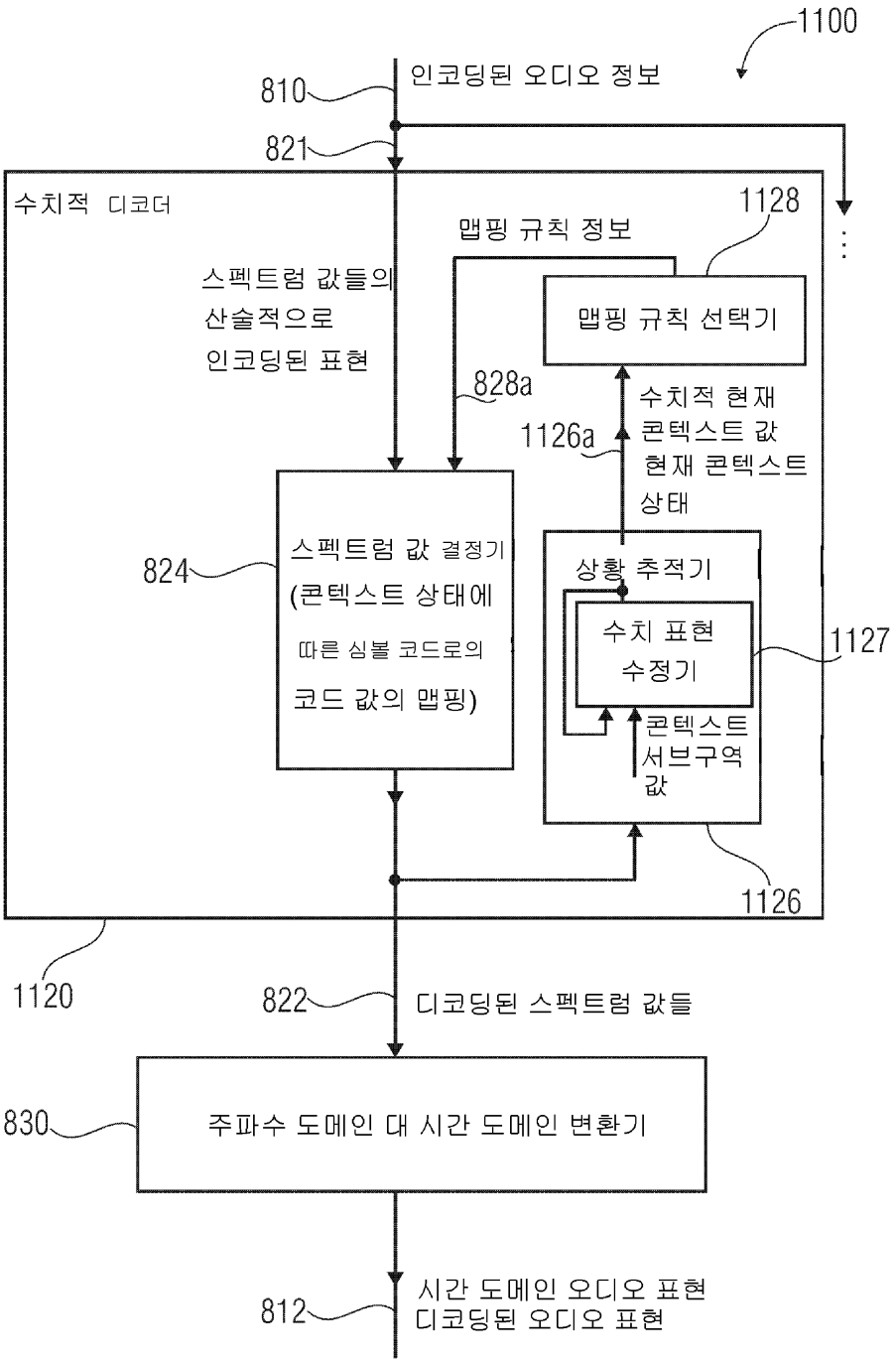
도면9


914
범례:
916
비 유효 상태에 대한 맵핑 규칙 인덱스 값을 가리키는 표시
개개의 유효 상태에 대한 맵핑 규칙 인덱스 값을 가리키는 표시
부적절한 유효 상태에 대한 맵핑 규칙 인덱스 값을 가리키는 표시
맵핑 규칙 인덱스 값
912
해시 테이블 엔트리 (c1,mriv1)
해시 테이블 엔트리(c2 ,mriv2)
ari_hash_m[i1]
ari_hash_m[i2]
ari_lookup_m[i2-1]
ari_lookup_m[i2]
ari_lookup_m[i1]
mriv1
mriv3
mriv2
mriv4
ari_lookup_m[i1-1]
910
구간
수치적 값
현재 콘텍스트 값
932
c1-1
c1
c1+1
934
c2-1
c2
c2+1
936

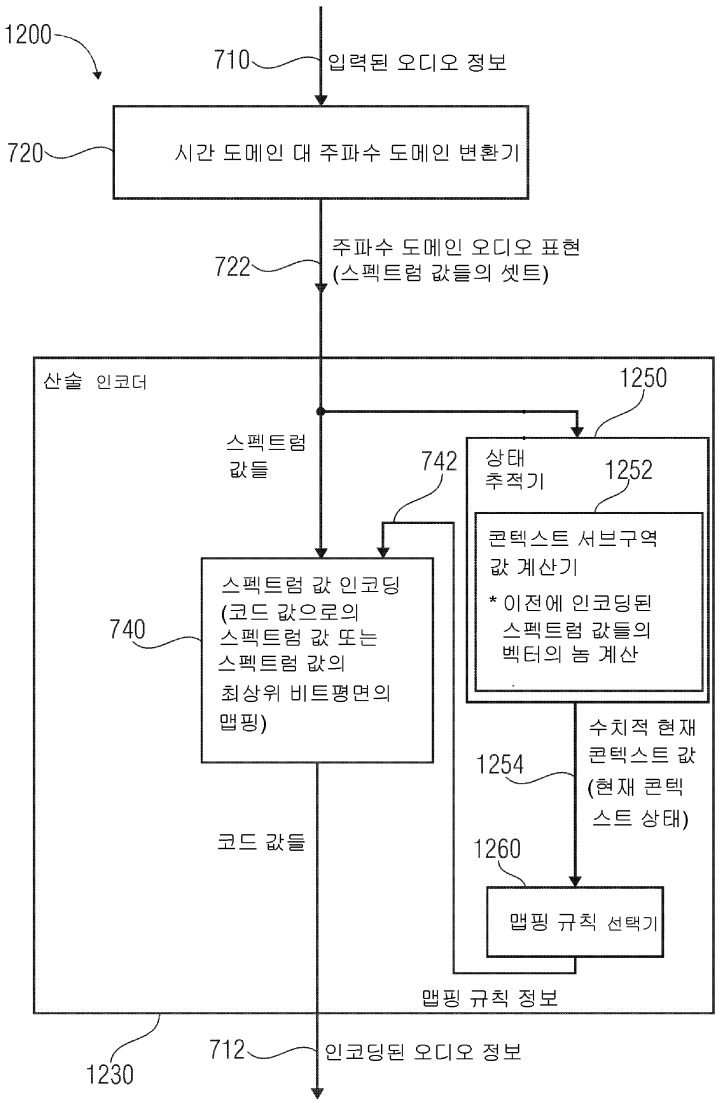
도면10


1000
710
입력된 오디오 정보
720
시간 도메인 대 주파수 도메인 변환기
주파수 도메인 오디오 표현
(스펙트럼 값들의 셋트)
722
콘텍스트
서브구역 값
산술 인코더
스펙트럼
값들
742
상태
추적기
수치 표현
수정기
스펙트럼 값 인코딩
(코드 값으로의
스펙트럼 값 또는
스펙트럼 값의
최상위 비트평면의
맵핑)
740
1052
1050
수치적 현재
콘텍스트
상태
754
(현재 콘텍
스트 상태)
코드 값들
맵핑
규칙 정보
1060
맵핑 규칙 선택기
1030
712
인코딩된 오디오 정보

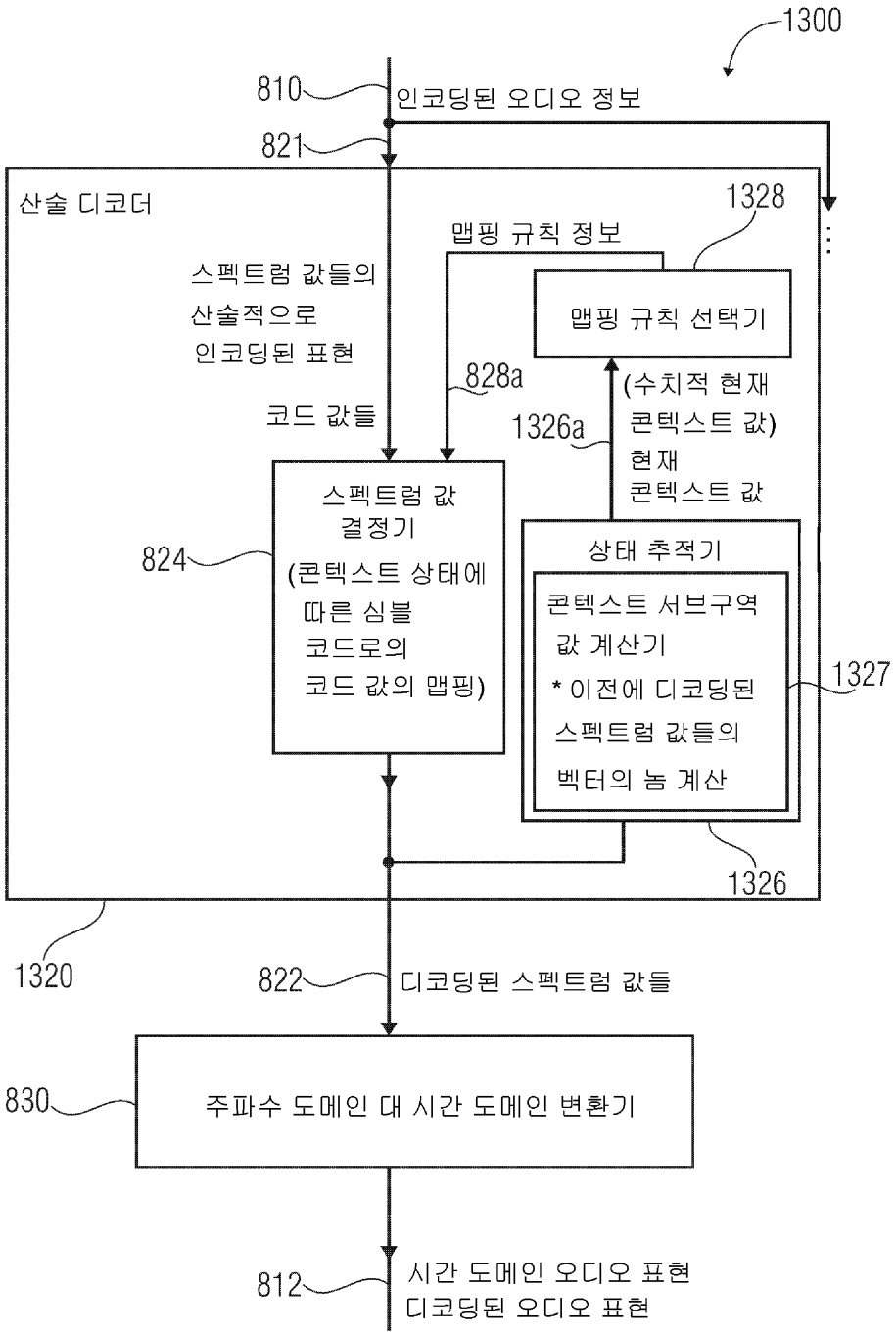
도면11


1100
810
인코딩된 오디오 정보
821
수치적 디코더
1128
맵핑 규칙 정보
스펙트럼 값들의
산술적으로
인코딩된 표현
맵핑 규칙 선택기
828a
1126a
수치적 현재
콘텍스트 값
현재 콘텍스트
상태
824
스펙트럼 값 결정기
(콘텍스트 상태에
따른 심볼 코드로의
코드 값의 맵핑)
상황 추적기
수치 표현
수정기
1127
콘텍스트
서브구역
값
1126
1120
822
디코딩된 스펙트럼 값들
830
주파수 도메인 대 시간 도메인 변환기
812
시간 도메인 오디오 표현
디코딩된 오디오 표현

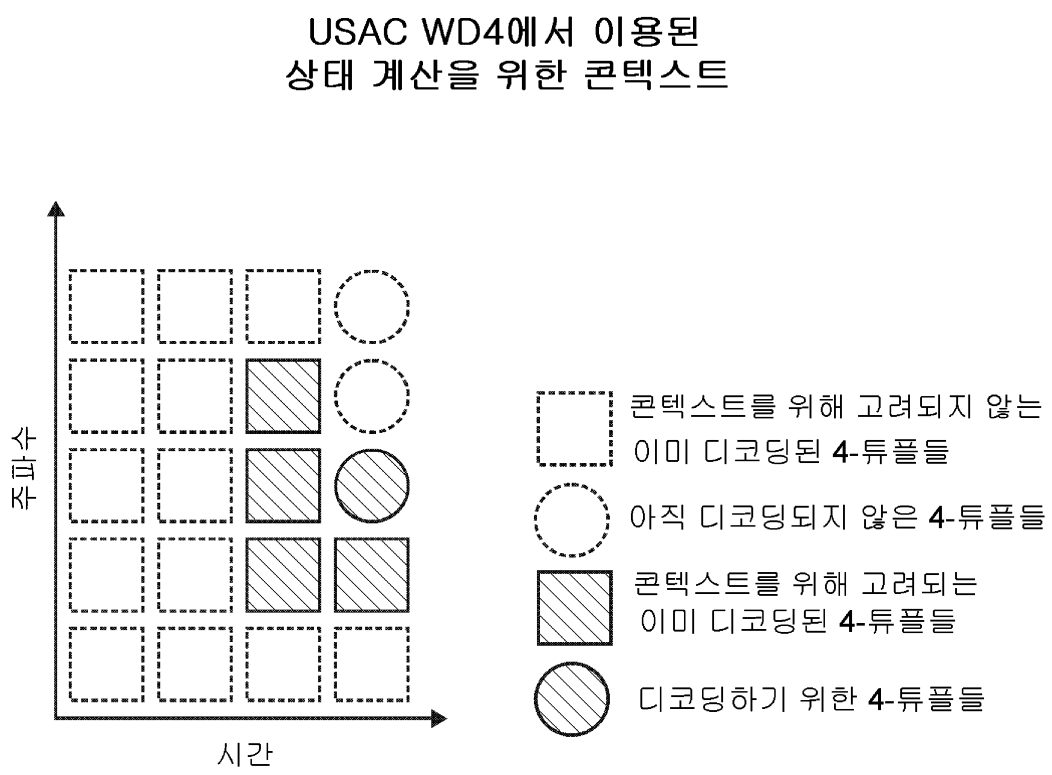
도면12


1200
710
입력된 오디오 정보
720
시간 도메인 대 주파수 도메인 변환기
722
주파수 도메인 오디오 표현
(스펙트럼 값들의 셋트)
산술 인코더
1250
스펙트럼
값들
742
상태
추적기
1252
콘텍스트 서브구역
값 계산기
* 이전에 인코딩된
스펙트럼 값들의
벡터의 놈 계산
스펙트럼 값 인코딩
(코드 값으로의
스펙트럼 값 또는
스펙트럼 값의
최상위 비트평면의
맵핑)
740
수치적 현재
콘텍스트 값
(현재 콘텍
스트 상태)
1254
코드 값들
1260
맵핑 규칙 선택기
맵핑 규칙 정보
712
인코딩된 오디오 정보
1230

도면13


1300
810
인코딩된 오디오 정보
821
산술 디코더
1328
맵핑 규칙 정보
맵핑 규칙 선택기
스펙트럼 값들의
산술적으로
인코딩된 표현
828a
(수치적 현재
콘텍스트 값)
1326a
현재
콘텍스트 값
코드 값들
스펙트럼 값
결정기
824
(콘텍스트 상태에
따른 심볼
코드로의
코드 값의 맵핑)
상태 추적기
콘텍스트 서브구역
값 계산기
* 이전에 디코딩된
스펙트럼 값들의
벡터의 놈 계산
1327
1326
1320
822
디코딩된 스펙트럼 값들
830
주파수 도메인 대 시간 도메인 변환기
812
시간 도메인 오디오 표현
디코딩된 오디오 표현

도면14a

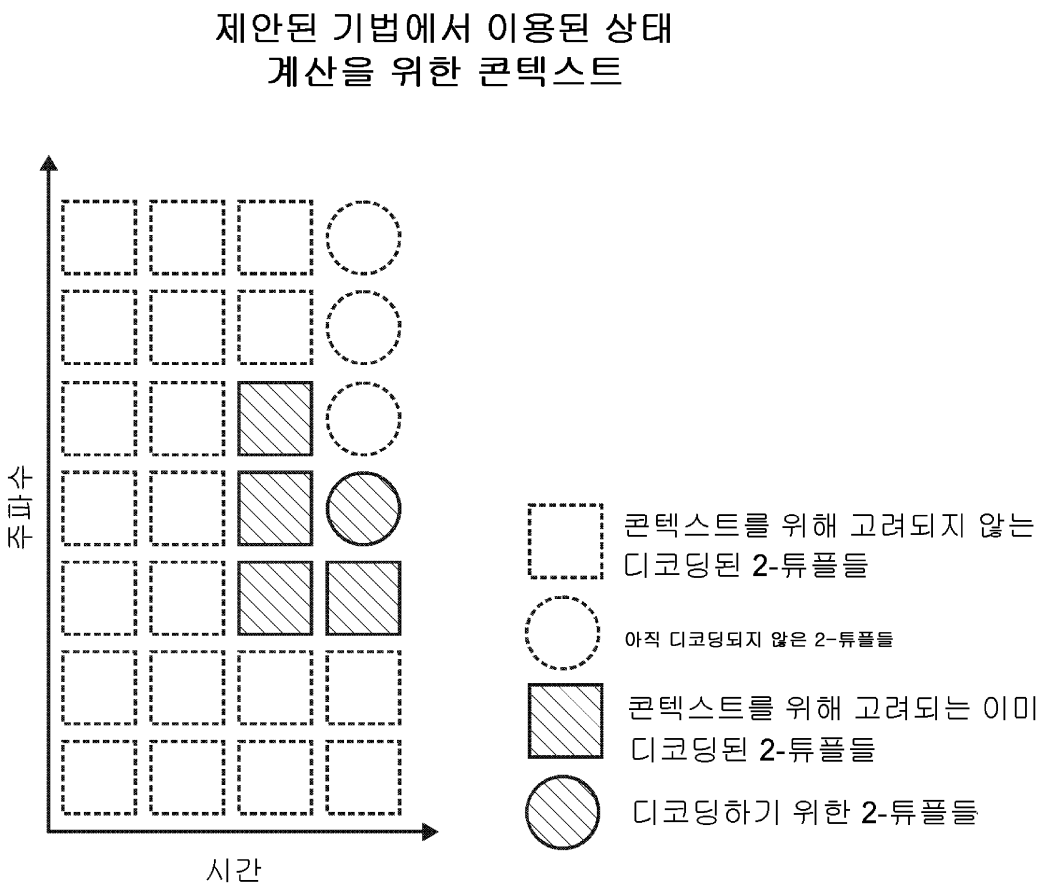


도면14b

USAC WD4 산술 코딩 기법에서 이용된 테이블

테이블 명칭	설명	데이터 유닛	메모리 (32 비트 워드)
arith_cf_ng_hash[128]	확률 모델 인덱스에 콘텍스트를 맵핑하는 해시 테이블	워드	128
arith_cf_ng[32][545]	각각의 확률 분포 모델에 대한 그룹들의 누적 빈도	1/2 워드	8720
egroups[8][8][8][8]	4-튜플의 그룹 인덱스	1/2 워드	2048
dgvectors[4*4096]	4-튜플에 그룹 인덱스 및 성분 인덱스를 맵핑한다	1/4 워드	4096
dgroups[544]	그룹의 기수(cardinal)에 그룹 인덱스를 맵핑하고 dg벡터들로 오프셋(offset)한다	워드	544
arith_cf_ne[2701]	성분 인덱스 심볼의 누적 빈도	1/2 워드	1350.5
arith_cf_r[16]	최하위 비트 평면들의 누적 빈도	1/2 워드	8
합계			16894.5

도면15a



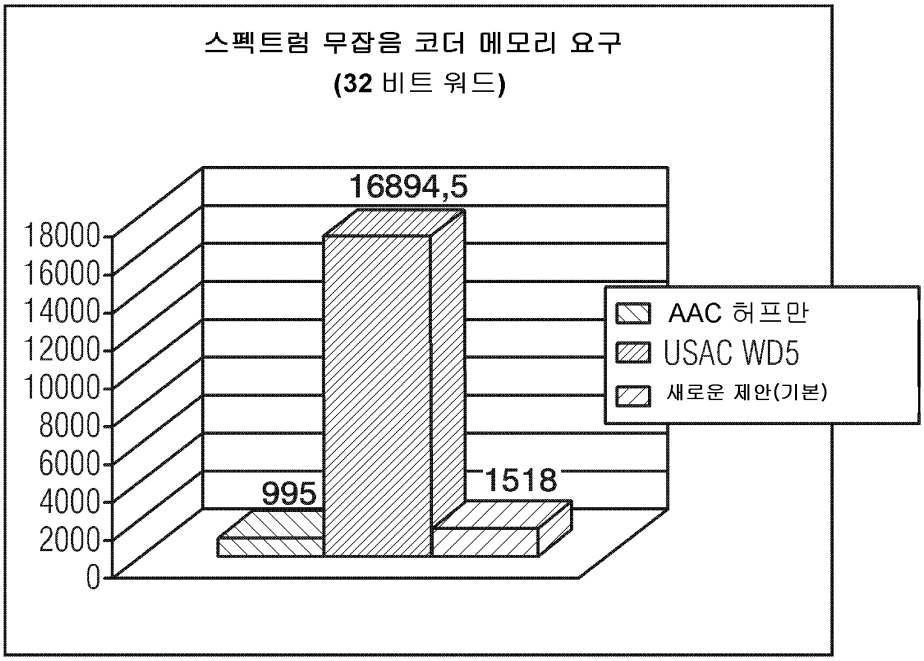
도면15b

제안된 코딩 기법에서 이용된 테이블

테이블 명칭	설명	데이터 유닛	메모리 (32 비트 워드)
arith_hash[600]	상태들의 그룹에 콘텍스트의 상태들을 맵핑하는 해시 테이블	1 워드	600
arith_lookup[600]	검색 테이블 맵핑	1/4 워드	150
arith_cf_msb[96][16]	누적 빈도 테이블에 대한 상태들의 그룹들 최상위 2 비트 방식 평면 m 및 ARITH_ESCAPE 심볼에 대한 누적 빈도들의 모델들	1/2 워드	768
합계			**1518**

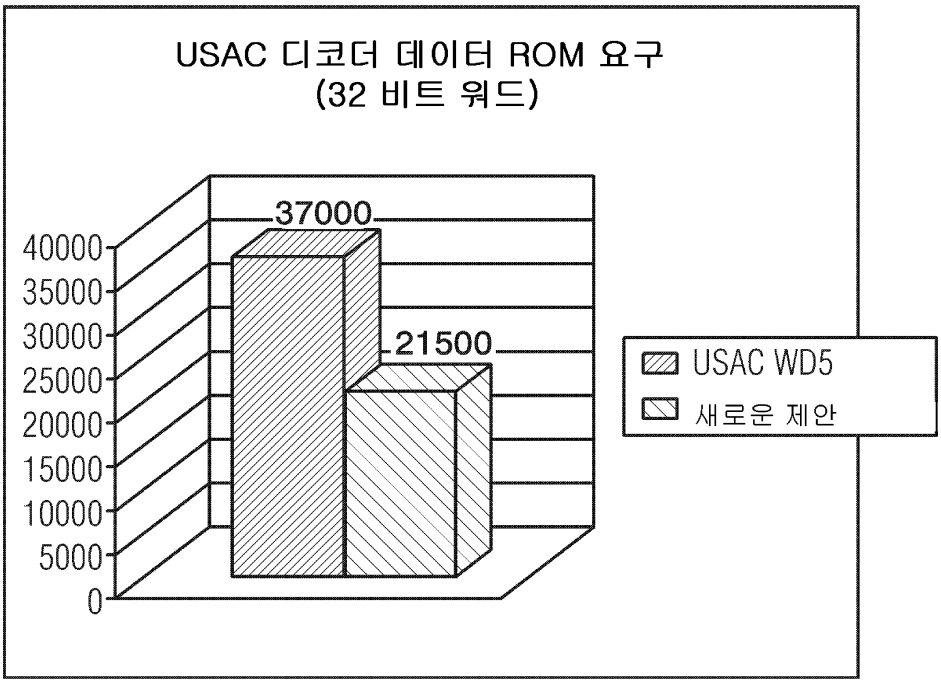
도면16a

제안된 무잡음 코딩 기법과
WD4에서의 ROM 요구



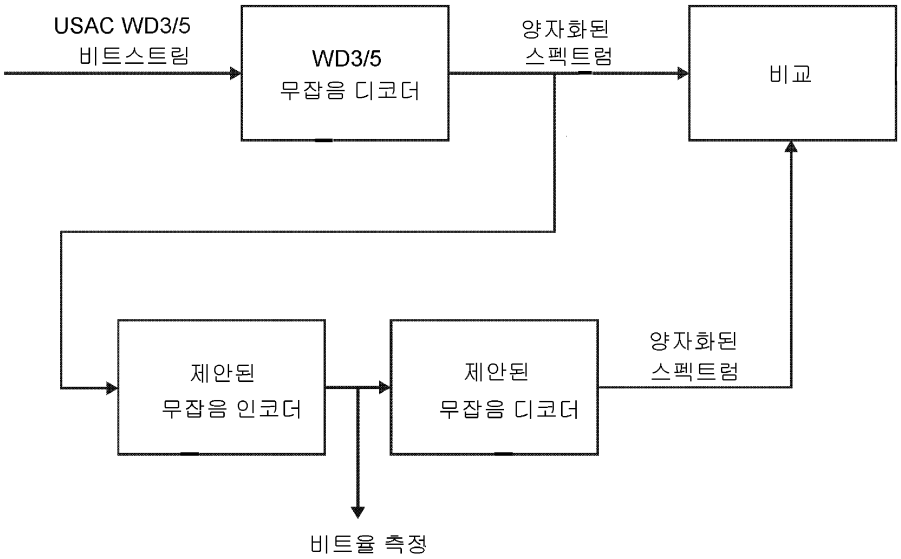
도면16b

WD4 및 제안된 기법에서의 전체
USAC 디코더 데이터 ROM 요구



도면17

제안된 코딩 기법과 WD3/WD5 무잡음 코딩의 비교



도면18

테이블: WD3 산술 코더 및 새로운 제안에 의해 생기는 평균 비트율. 순(net) 비트율은 바이트 정렬이나 채움(fill) 비트를 위한 비트를 포함하지 않는다.

연산모드	WD3 (kbit/s)	새로운 제안 (kbit/s)	트랜스코딩 이후의 차이 (kbit/s)	트랜스코딩 이후의 차이 (전체 비트율 %)
Test 1, 64s	64.00	62.78	-1.22	-1.90
Test 2, 32s	32.00	31.47	-0.53	-1.66
Test 3, 24s	24.00	23.61	-0.39	-1.64
Test 4, 20s	20.00	19.65	-0.35	-1.74
Test 5, 16s	16.00	15.72	-0.28	-1.75
Test 6, 24m	24.00	23.56	-0.44	-1.83
Test 7, 20m	20.00	19.61	-0.39	-1.95
Test 8, 16m	16.00	15.70	-0.30	-1.87
Test 9, 12m	12.00	11.77	-0.23	-1.92

도면19

테이블: WD3 산술 코더 및 제안에 대한
최소 및 최대 비트저장 레벨

연산모드	비트저장 제어					
	새로운 제안			WD3		
	min	max	avg	min	max	avg
Test 1, 64kbps stereo	3739	9557	8874	2314	9557	7018
Test 2, 32kbps stereo	2335	4505	4293	582	4505	3529
Test 3, 24kbps stereo	2184	4704	4472	957	4704	3871
Test 4, 20kbps stereo	2688	4864	4660	712	4864	3854
Test 5, 16kbps stereo	2965	5006	4859	724	5006	4234
Test 6, 24kbps mono	2185	4704	4457	1002	4704	3927
Test 7, 20kbps mono	2782	4864	4690	1192	4864	3935
Test 8, 16kbps mono	2916	5006	4905	1434	5006	4450
Test 9, 12kbps mono	3645	5184	5107	2256	5184	4787

도면20

테이블: 산술 코더의 각각 다른 버전에 대하여
32 kbit/s WD3 비트스트림을 디코딩하기 위한
평균 복잡도 수치

	WD3	기본 버전
PCU (MHz)	0.953	0.823

도면21a

```
static unsigned short ari_lookup_m[600] = {
0x02,0x01,0x03,0x38,0x3C,0x44,0x45,0x05,
0x07,0x37,0x41,0x08,0x0A,0x07,0x38,0x44,
0x0B,0x14,0x3E,0x3B,0x0C,0x14,0x3E,0x0C,
0x2B,0x5F,0x0F,0x42,0x3B,0x44,0x11,0x36,
0x42,0x41,0x13,0x2B,0x3E,0x3B,0x0C,0x14,
0x3E,0x14,0x2B,0x3E,0x15,0x07,0x3B,0x11,
0x25,0x42,0x41,0x16,0x36,0x3A,0x41,0x25,
0x36,0x3A,0x14,0x3E,0x36,0x42,0x36,0x3A,
0x2B,0x3E,0x42,0x38,0x41,0x17,0x19,0x42,
0x3B,0x44,0x19,0x1C,0x42,0x3B,0x44,0x1D,
0x2B,0x38,0x12,0x2B,0x1E,0x20,0x42,0x41,
0x22,0x36,0x37,0x41,0x24,0x36,0x42,0x41,
0x26,0x07,0x3A,0x2B,0x3E,0x01,0x3E,0x27,
0x07,0x37,0x44,0x3F,0x29,0x42,0x3B,0x44,
0x2A,0x07,0x37,0x41,0x29,0x07,0x38,0x2B,
0x3E,0x36,0x3E,0x39,0x42,0x34,0x07,0x3B,
0x26,0x36,0x42,0x41,0x36,0x42,0x3B,0x36,
0x3A,0x02,0x3A,0x03,0x38,0x07,0x37,0x07,
0x37,0x39,0x3A,0x3F,0x3B,0x41,0x44,0x57,
0x2C,0x07,0x3C,0x03,0x31,0x42,0x3C,0x22,
0x36,0x38,0x2A,0x07,0x27,0x2E,0x07,0x38,
0x44,0x2F,0x07,0x37,0x41,0x03,0x32,0x42,
0x3B,0x44,0x32,0x07,0x38,0x26,0x07,0x3A,
0x26,0x3E,0x02,0x2A,0x07,0x3B,0x33,0x03,
0x42,0x3B,0x44,0x1B,0x36,0x42,0x41,0x32,
0x07,0x3B,0x26,0x36,0x42,0x3B,0x36,0x3E,
0x39,0x42,0x39,0x42,0x3B,0x26,0x07,0x42,
0x41,0x36,0x42,0x37,0x41,0x36,0x42,0x3B,
0x07,0x3A,0x03,0x3B,0x07,0x42,0x3C,0x39,
0x42,0x3C,0x07,0x38,0x07,0x38,0x37,0x38,
0x3C,0x41,0x44,0x57,0x3B,0x3A,0x33,0x37,
0x33,0x03,0x02,0x33,0x07,0x3C,0x33,0x07,
0x3B,0x2A,0x34,0x42,0x41,0x34,0x07,0x38,
0x36,0x3E,0x26,0x07,0x3C,0x39,0x42,0x41,
0x33,0x36,0x42,0x3C,0x26,0x07,0x42,0x41,
0x07,0x38,0x07,0x3A,0x39,0x37,0x39,0x42,
0x3B,0x26,0x07,0x37,0x41,0x01,0x07,0x42,
0x3C,0x39,0x42,0x3B,0x03,0x3F,0x03,0x3B,
```

도면21b

```
0x39,0x42,0x3B,0x39,0x37,0x3C,0x39,0x37,
0x3C,0x03,0x38,0x3A,0x3B,0x3C,0x41,0x44,
0x57,0x03,0x33,0x03,0x26,0x37,0x34,0x42,
0x41,0x39,0x3F,0x34,0x42,0x39,0x37,0x39,
0x42,0x3B,0x26,0x07,0x37,0x41,0x07,0x3B,
0x07,0x3A,0x03,0x42,0x41,0x39,0x42,0x3C,
0x39,0x42,0x3C,0x07,0x3F,0x03,0x3B,0x03,
0x3B,0x39,0x37,0x3C,0x02,0x42,0x3C,0x03,
0x3D,0x40,0x3C,0x41,0x44,0x57,0x03,0x39,
0x3D,0x03,0x3B,0x39,0x42,0x41,0x39,0x3A,
0x03,0x3F,0x39,0x3F,0x02,0x3E,0x3F,0x02,
0x40,0x3C,0x41,0x44,0x27,0x03,0x03,0x03,
0x3D,0x3F,0x40,0x39,0x41,0x44,0x03,0x3D,
0x42,0x43,0x03,0x3C,0x44,0x03,0x3D,0x40,
0x3C,0x02,0x3C,0x44,0x3D,0x43,0x3C,0x02,
0x41,0x44,0x3D,0x43,0x3C,0x03,0x44,0x3D,
0x43,0x41,0x02,0x44,0x3D,0x41,0x44,0x43,
0x41,0x44,0x43,0x41,0x44,0x03,0x3C,0x44,
0x46,0x47,0x49,0x4B,0x50,0x4D,0x4F,0x0B,
0x56,0x0C,0x0C,0x2B,0x03,0x52,0x0D,0x10,
0x1A,0x56,0x56,0x57,0x58,0x58,0x0C,0x26,
0x01,0x01,0x02,0x3D,0x1A,0x1E,0x56,0x5A,
0x5B,0x5B,0x00,0x27,0x15,0x0C,0x39,0x39,
0x26,0x01,0x02,0x02,0x3D,0x57,0x57,0x27,
0x57,0x00,0x39,0x39,0x02,0x02,0x03,0x27,
0x27,0x02,0x27,0x02,0x02,0x3D,0x5E,0x5D,
0x5F,0x5E,0x5D,0x5D,0x5D,0x5D,0x5C,0x5C,
0x5F,0x5D,0x5D,0x5D,0x5E,0x5D,0x5D,0x5C,
0x5C,0x5C,0x5C,0x5F,0x5D,0x5D,0x5D,0x5E,
0x5D,0x5C,0x5C,0x5E,0x5C,0x5E,0x5C,0x5C,
0x5F,0x5D,0x5C,0x5D,0x5F,0x5D,0x5D,0x5F,
0x5E,0x5F,0x5D,0x5F,0x5D,0x5F,0x5F,0x5D,
0x5F,0x5D,0x5F,0x5D,0x5F,0x5F,0x5F,0x5F,
0x5F,0x5F,0x5F,0x5F,0x44,0x5F,0x5E,0x5D,
0x5D,0x5C,0x5C,0x5D,0x5C,0x5C,0x5C,0x5F,
0x5E,0x5D,0x5E,0x5D,0x5E,0x5D,0x5E,0x5F,
0x5E,0x5F,0x5E,0x5C,0x5E,0x5C,0x5E,0x5E,

};
```

도면22a

```
static unsigned long ari_hash_m[600] = {
0x00000100UL,0x00000302UL,0x0000053AUL,0x0000073BUL,0x00000A41UL,0x00000F44UL,
0x00111104UL,0x00111306UL,
0x00111542UL,0x0011173BUL,0x00111F44UL,0x00112209UL,0x0011242BUL,0x0011263EUL,
0x0011293CUL,0x00113105UL,
0x0011330CUL,0x0011352BUL,0x0011383AUL,0x00113F44UL,0x0011440CUL,0x0011462BUL,
0x00114F41UL,0x0011530CUL,
0x0012110DUL,0x0012120EUL,0x00121436UL,0x00121637UL,0x00121941UL,0x00122110UL,
0x00122312UL,0x00122507UL,
0x00122738UL,0x00123156UL,0x00123314UL,0x00123507UL,0x0012373AUL,0x00123F44UL,
0x00124212UL,0x0012452BUL,
0x00124F44UL,0x00125314UL,0x0012762BUL,0x00131121UL,0x00131323UL,0x00131542UL,
0x0013211DUL,0x00132216UL,
0x00132436UL,0x00132738UL,0x0013311DUL,0x00133329UL,0x00133507UL,0x00133838UL,
0x00134115UL,0x0013432BUL,
0x0013463EUL,0x00134F44UL,0x0013532BUL,0x0013FF3BUL,0x00142307UL,0x00143126UL,
0x00143407UL,0x00143E44UL,
0x00144307UL,0x0014FF3BUL,0x0015633EUL,0x0017863BUL,0x0021005AUL,0x00211218UL,
0x00211407UL,0x00211637UL,
0x00211941UL,0x0021211AUL,0x0021221BUL,0x00212436UL,0x00212637UL,0x00212941UL,
0x00213118UL,0x00213320UL,
0x00213507UL,0x00213F44UL,0x00214314UL,0x00220001UL,0x0022121FUL,0x00221407UL,
0x0022173BUL,0x00222121UL,
0x00222323UL,0x00222542UL,0x0022283BUL,0x0022311DUL,0x00223325UL,0x00223507UL,
0x00223737UL,0x00224115UL,
0x00224436UL,0x0022463EUL,0x00224F44UL,0x00225436UL,0x00225F44UL,0x0022622BUL,
0x0023112DUL,0x00231330UL,
0x00231542UL,0x0023183CUL,0x0023212DUL,0x00232228UL,0x00232407UL,0x00232637UL,
0x00232941UL,0x00233115UL,
0x0023332BUL,0x00233542UL,0x0023383BUL,0x0023412AUL,0x00234336UL,0x00234642UL,
0x00234F44UL,0x00235407UL,
0x00235F44UL,0x0023653EUL,0x0023FF3BUL,0x00241307UL,0x00241F44UL,0x00242336UL,
0x00242542UL,0x00242F44UL,
0x00243234UL,0x00243407UL,0x00243738UL,0x00244126UL,0x00244307UL,0x0024463AUL,
0x00244F44UL,0x00245442UL,
0x00245F44UL,0x00246236UL,0x0024FF3CUL,0x00252442UL,0x00252F44UL,0x00253303UL,
0x00253F44UL,0x00254303UL,
0x00254F44UL,0x00255303UL,0x0025FF41UL,0x0026733EUL,0x0027FF44UL,0x002AF203UL,
0x002FFF44UL,0x0031115BUL,
0x00311331UL,0x00311542UL,0x00312121UL,0x0031222DUL,0x00312436UL,0x00312637UL,
0x00312F44UL,0x00313335UL,
```

도면22b

```
0x00313507UL,0x00313F44UL,0x00314332UL,0x0031FF3CUL,0x0032112CUL,0x00321335UL,
0x00321542UL,0x0032183CUL,
0x0032212CUL,0x00322330UL,0x00322542UL,0x0032273BUL,0x0032312DUL,0x00323231UL,
0x00323407UL,0x00323637UL,
0x00323A41UL,0x0032412AUL,0x00324407UL,0x00324642UL,0x00324F44UL,0x00325336UL,
0x00325507UL,0x00325F44UL,
0x00326234UL,0x0032FF3CUL,0x00331127UL,0x00331332UL,0x00331542UL,0x0033212EUL,
0x00332334UL,0x00332407UL,
0x00332637UL,0x00332941UL,0x00333115UL,0x00333235UL,0x00333407UL,0x00333738UL,
0x0033412AUL,0x00334336UL,
0x00334637UL,0x00334F44UL,0x00335234UL,0x00335407UL,0x0033573AUL,0x00335F44UL,
0x00336407UL,0x0033FF3CUL,
0x00341307UL,0x00341F44UL,0x00342307UL,0x00342542UL,0x00342F44UL,0x00343234UL,
0x00343442UL,0x0034373BUL,
0x00344126UL,0x00344307UL,0x00344542UL,0x0034483BUL,0x00345126UL,0x00345307UL,
0x00345637UL,0x00345F44UL,
0x00346442UL,0x0034FF3CUL,0x00352442UL,0x00353139UL,0x00353303UL,0x00353637UL,
0x00353F44UL,0x00354303UL,
0x00354637UL,0x00354F44UL,0x00355442UL,0x00356102UL,0x00356303UL,0x0035FF41UL,
0x00366203UL,0x0037733DUL,
0x0039E43AUL,0x003BF641UL,0x003FFF44UL,0x00411227UL,0x00411334UL,0x0041212CUL,
0x00412407UL,0x0041312EUL,
0x00413334UL,0x0041FF3CUL,0x00421127UL,0x00421334UL,0x00421542UL,0x00422127UL,
0x00422334UL,0x00422542UL,
0x00422F44UL,0x00423232UL,0x00423407UL,0x0042373BUL,0x00424126UL,0x00424336UL,
0x00424542UL,0x00424F44UL,
0x00425407UL,0x0042FF3CUL,0x00431339UL,0x00431542UL,0x00432133UL,0x00432407UL,
0x0043273BUL,0x00432F44UL,
0x00433234UL,0x00433407UL,0x00433637UL,0x00433F44UL,0x00434234UL,0x00434442UL,
0x00434738UL,0x00435126UL,
0x00435542UL,0x00435F44UL,0x00436442UL,0x0043FF41UL,0x00441303UL,0x00442126UL,
0x00442307UL,0x00442542UL,
0x00442F44UL,0x00443239UL,0x00443442UL,0x0044373BUL,0x00443F44UL,0x00444234UL,
0x00444442UL,0x00444637UL,
0x00444F44UL,0x00445307UL,0x00445637UL,0x00445F44UL,0x00446537UL,0x0044FF41UL,
0x00452442UL,0x00452F44UL,
0x00453303UL,0x00453537UL,0x00453F44UL,0x00454303UL,0x00454638UL,0x00454F44UL,
0x00455303UL,0x00455638UL,
0x00455F44UL,0x00456437UL,0x0045FF41UL,0x00466203UL,0x00478438UL,0x0049FF44UL,
0x004CF741UL,0x004FFF44UL,
```

도면22c

```
0x00511226UL,0x00512127UL,0x00512339UL,0x00521127UL,0x00521339UL,0x00522127UL,
0x00522339UL,0x00522637UL,
0x00523126UL,0x00523403UL,0x00524126UL,0x00524307UL,0x00531126UL,0x00531307UL,
0x00532126UL,0x00532307UL,
0x00532537UL,0x00532F44UL,0x00533239UL,0x00533442UL,0x0053373BUL,0x00534126UL,
0x00534542UL,0x00534F44UL,
0x00535442UL,0x0053FF3CUL,0x00542303UL,0x00542638UL,0x00543102UL,0x00543307UL,
0x00543638UL,0x00543F44UL,
0x00544307UL,0x00544537UL,0x00545102UL,0x00545303UL,0x0054FF41UL,0x00552437UL,
0x00552F44UL,0x00553437UL,
0x00553F44UL,0x00554303UL,0x0055463BUL,0x00554F44UL,0x00555307UL,0x00555537UL,
0x00556102UL,0x00556342UL,
0x00566203UL,0x00577342UL,0x0059733AUL,0x005CF53CUL,0x005FFF44UL,0x00611239UL,
0x00621127UL,0x00623307UL,
0x00631126UL,0x00632442UL,0x00632F44UL,0x00633303UL,0x00633638UL,0x00634102UL,
0x00634303UL,0x0063FF41UL,
0x00642303UL,0x00642F44UL,0x00643303UL,0x00643F44UL,0x00644207UL,0x00655203UL,
0x00665F44UL,0x00666303UL,
0x00677203UL,0x0069A53BUL,0x006DF841UL,0x006FFF44UL,0x00711239UL,0x00721127UL,
0x00722239UL,0x00733239UL,
0x00744437UL,0x00755203UL,0x00776F44UL,0x00777437UL,0x007BF33FUL,0x007FFF44UL,
0x00822239UL,0x00833203UL,
0x00854540UL,0x00888102UL,0x00888538UL,0x008BF23DUL,0x008FFF44UL,0x00922239UL,
0x0094333DUL,0x0096533DUL,
0x00998F44UL,0x00999303UL,0x009BF53BUL,0x009FFF44UL,0x00A44203UL,0x00A6633FUL,
0x00AA9F44UL,0x00AAA303UL,
0x00ADF33DUL,0x00AFFF44UL,0x00B33203UL,0x00B55440UL,0x00BBAC44UL,0x00BBB53BUL,
0x00BFFF44UL,0x00C33203UL,
0x00C6423DUL,0x00CCBF44UL,0x00CCC303UL,0x00CFFF44UL,0x00D4423DUL,0x00DDD303UL
0x00DFFF44UL,0x00E6453CUL,
0x00EEE342UL,0x00EFFF44UL,0x00F55343UL,0x00F7FF44UL,0x00FFEF44UL,0x00FFF33DUL,
0x00FFF744UL,0x01000145UL,
0x01011147UL,0x01111248UL,0x0111224AUL,0x0111324DUL,0x0112114CUL,0x0112214EUL,
0x01123108UL,0x01131150UL,
0x01132150UL,0x01133150UL,0x01141126UL,0x01144100UL,0x01211151UL,0x01212153UL,
0x01213108UL,0x01221154UL,
0x01222155UL,0x01223117UL,0x01224212UL,0x01231215UL,0x01232215UL,0x01233215UL,
0x01241126UL,0x01242239UL,
0x0124322BUL,0x01251103UL,0x01255100UL,0x01311159UL,0x01312155UL,0x01313117UL,
0x01315201UL,0x0132122CUL,
```

도면22d

```
0x0132222CUL,0x0132321DUL,0x01331157UL,0x01332158UL,0x01333150UL,0x01341126UL,
0x01342127UL,0x01343100UL,
0x01344100UL,0x01351103UL,0x01355100UL,0x01369100UL,0x0141115AUL,0x01421227UL,
0x01422227UL,0x01431226UL,
0x01432226UL,0x01441127UL,0x01442127UL,0x01443100UL,0x01444100UL,0x01458100UL,
0x01511157UL,0x01531239UL,
0x01555100UL,0x01611157UL,0x01631239UL,0x01777100UL,0x01D33102UL,0x01FFF203UL,
0x0200055DUL,0x0200085DUL,
0x02000F5FUL,0x0211155DUL,0x02111F44UL,0x02112F5FUL,0x02121F44UL,0x02122F44UL,
0x02133F5FUL,0x0217FF5FUL,
0x021FFF44UL,0x02211F44UL,0x02212F44UL,0x02221F44UL,0x0222245EUL,0x02222F5FUL,
0x02223F5FUL,0x02232F5FUL,
0x02233F5FUL,0x02243F5FUL,0x022BFF5FUL,0x022FFF44UL,0x02322F44UL,0x02323F5FUL,
0x02332F5FUL,0x0233355CUL,
0x02333F5FUL,0x02334F5FUL,0x0233FF5CUL,0x02343F5FUL,0x0234FF5CUL,0x02353F5FUL,
0x02364F5FUL,0x0239655CUL,
0x023FFF44UL,0x02433F5FUL,0x0246445CUL,0x024A955DUL,0x024FFF44UL,0x0257435EUL,
0x025AC55CUL,0x025FFF44UL,
0x0267565DUL,0x026FFF44UL,0x0278655DUL,0x027FFF44UL,0x0288875DUL,0x028CC95DUL,
0x028FFF44UL,0x029A965DUL,
0x029FFF44UL,0x02ABA65DUL,0x02AFFF44UL,0x02BAFF5FUL,0x02BFFF44UL,0x02CCC45DUL,
0x02CFFF44UL,0x02DFFF44UL,
0x02EEE65DUL,0x02EFFF44UL,0x02F7335EUL,0x02FF0044UL,0x02FFEF44UL,0x02FFFF44UL,
0x0400045EUL,0x04000F5FUL,
0x04111F5DUL,0x04234F5DUL,0x042DFF5CUL,0x04343F5DUL,0x043FFF5FUL,0x044FFF5FUL,
0x045ED75CUL,0x045FFF5FUL,
0x046DFF5FUL,0x046FFF5FUL,0x047B5C5CUL,0x047FFF5FUL,0x048B435EUL,0x048FFF5FUL,
0x0499FF5CUL,0x04EFFF5FUL,
0x04FDFE5DUL,0x04FFFF5FUL,0x0600075EUL,0x06DFFF5FUL,0x06FFFF5FUL,0x08BFFF5CUL,
0x08FFFF5CUL,0x7FFFFFF4FUL,

};
```

도면23a

```
unsigned short ari_cf_m [96][17] = {
{ 7529, 5263, 5173, 5162, 2636, 825, 674, 653, 511, 281, 210, 195, 173, 130, 105,
96,
  0
},
{ 10351, 7392, 7203, 7178, 4327, 1620, 1279, 1230, 1008, 606, 436, 399, 362, 288, 233,
212,
  0
},
{ 12505, 9566, 9216, 9157, 6631, 3220, 2541, 2418, 2086, 1404, 1024, 922, 858, 721,
597, 535,
  0
},
{ 14710,12600,11956,11801,10114, 7056, 5723, 5381, 4870, 3724, 2870, 2566, 2437, 2122,
1809, 1609,
  0
},
{ 4186, 2623, 2608, 2607, 939, 109, 86, 84, 63, 23, 14, 13, 11, 8, 5, 4,
  0
},
{ 7310, 4598, 4547, 4544, 2079, 354, 264, 259, 204, 90, 42, 38, 34, 25, 14, 11,
  0
},
{ 9998, 6785, 6641, 6630, 4087, 1058, 770, 746, 621, 339, 166, 143, 131, 104, 67,
49,
  0
},
{ 13801,11125,10550,10472, 8498, 5030, 3694, 3509, 3101, 2141, 1247, 1038, 975, 821,
623, 440,
  0
},
{ 5784, 3557, 3523, 3521, 1359, 164, 122, 119, 85, 29, 16, 14, 12, 8, 5, 4,
  0
},
{ 7617, 4716, 4649, 4645, 2129, 361, 258, 252, 191, 80, 39, 34, 30, 22, 13, 10,
  0
},
{ 9553, 6223, 6064, 6052, 3524, 912, 643, 620, 501, 262, 132, 112, 102, 79, 52, 39,
  0
},
{ 8423, 5300, 5186, 5179, 2629, 539, 375, 362, 276, 119, 61, 53, 46, 33, 21, 17,
  0
},
{ 9570, 6162, 5952, 5936, 3574, 1016, 702, 670, 538, 282, 152, 129, 117, 90, 63,
50,
  0
},
```

2310

2312

도면23b

```
{ 7355, 4678, 4630, 4628, 1834, 252, 190, 187, 124, 38, 21, 19, 16, 10, 6, 5,
  0
},
{ 8916, 5717, 5627, 5622, 2647, 471, 340, 332, 242, 97, 46, 41, 36, 25, 15, 12,
  0
},
{ 10839, 7394, 7198, 7184, 4273, 1169, 832, 803, 643, 327, 167, 143, 130, 100, 66,
49,
  0
},
{ 8036, 5132, 5045, 5040, 2207, 350, 250, 243, 167, 56, 31, 28, 23, 15, 10, 8,
  0
},
{ 9663, 6256, 6110, 6101, 3064, 633, 441, 426, 311, 127, 65, 57, 49, 34, 22, 18,
  0
},
{ 11237, 7629, 7361, 7339, 4460, 1302, 898, 859, 675, 333, 178, 152, 136, 104, 73,
57,
  0
},
{ 10436, 6930, 6718, 6703, 3735, 902, 619, 595, 446, 189, 98, 85, 74, 51, 34, 28,
  0
},
{ 11678, 7933, 7574, 7541, 4826, 1560, 1057, 1002, 796, 406, 218, 184, 165, 124, 88,
70,
  0
},
{ 10597, 7265, 7057, 7039, 3904, 1071, 768, 738, 541, 227, 127, 111, 93, 60, 40,
34,
  0
},
{ 11298, 7892, 7596, 7566, 4416, 1276, 873, 832, 611, 261, 141, 122, 103, 70, 47,
40,
  0
},
{ 6435, 4238, 4210, 4208, 1558, 216, 174, 172, 116, 39, 24, 22, 18, 11, 7, 6,
  0
},
{ 8744, 5744, 5671, 5667, 2636, 496, 375, 368, 273, 113, 58, 52, 45, 32, 19, 15,
  0
},
{ 10899, 7632, 7457, 7444, 4488, 1270, 942, 914, 737, 383, 204, 177, 161, 122, 80,
62,
  0
},
```

도면23c

```
{ 7929, 5176, 5119, 5116, 2169, 307, 230, 226, 155, 51, 28, 25, 21, 13, 8, 7,
  0
},
{ 9520, 6304, 6204, 6199, 3044, 585, 424, 415, 304, 118, 59, 52, 45, 31, 19, 15,
  0
},
{ 11187, 7805, 7600, 7586, 4554, 1292, 928, 896, 703, 342, 178, 153, 138, 103, 68,
52,
  0
},
{ 10371, 7037, 6881, 6871, 3721, 860, 613, 596, 440, 176, 88, 77, 67, 46, 28, 23,
  0
},
{ 8562, 5804, 5738, 5734, 2472, 467, 374, 369, 233, 76, 48, 44, 34, 20, 13, 11,
  0
},
{ 10078, 6838, 6713, 6705, 3407, 782, 591, 578, 407, 164, 92, 83, 69, 46, 29, 24,
  0
},
{ 11694, 8342, 8108, 8089, 4999, 1559, 1143, 1105, 858, 424, 240, 210, 185, 135, 92,
73,
  0
},
{ 9115, 6191, 6084, 6077, 2924, 604, 455, 445, 304, 109, 64, 58, 47, 29, 20, 17,
  0
},
{ 10786, 7434, 7256, 7244, 3968, 1006, 734, 712, 517, 213, 118, 104, 88, 60, 40,
34,
  0
},
{ 12216, 8840, 8535, 8507, 5492, 1860, 1342, 1289, 1007, 505, 289, 249, 221, 163, 115,
94,
  0
},
{ 11473, 8085, 7843, 7825, 4602, 1304, 931, 898, 669, 283, 153, 135, 116, 80, 53,
44,
  0
},
{ 12650, 9279, 8908, 8872, 5924, 2160, 1539, 1470, 1162, 585, 329, 282, 251, 184, 130,
107,
  0
},
{ 12597, 9310, 8859, 8806, 6019, 2457, 1741, 1643, 1310, 684, 401, 343, 304, 225, 164,
137,
  0
},
```

도면23d

```
{ 11509, 8255, 8030, 8012, 4706, 1489, 1135, 1103, 793, 347, 212, 192, 154, 98, 68,
59,
  0
},
{ 11946, 8648, 8370, 8344, 5106, 1628, 1185, 1142, 842, 366, 206, 183, 153, 100, 68,
57,
  0
},
{ 13088, 9856, 9444, 9397, 6468, 2535, 1831, 1745, 1384, 700, 407, 352, 310, 224, 160,
132,
  0
},
{ 12262, 9021, 8683, 8647, 5519, 1929, 1405, 1342, 1024, 481, 280, 245, 210, 146, 104,
89,
  0
},
{ 13353,10225, 9742, 9678, 6908, 2945, 2133, 2017, 1619, 875, 524, 450, 401, 298, 218,
183,
  0
},
{ 10055, 6990, 6879, 6872, 3544, 882, 687, 675, 490, 209, 121, 112, 95, 63, 41, 36,
  0
},
{ 10647, 7427, 7278, 7270, 3911, 968, 723, 705, 513, 203, 109, 98, 83, 55, 34, 28,
  0
},
{ 11221, 8027, 7861, 7851, 4397, 1264, 981, 961, 689, 293, 174, 159, 132, 83, 54,
46,
  0
},
{ 11700, 8429, 8209, 8192, 4825, 1451, 1079, 1049, 767, 327, 186, 167, 140, 92, 61,
52,
  0
},
{ 12949, 9762, 9414, 9379, 6351, 2418, 1786, 1717, 1351, 686, 398, 345, 301, 216, 152,
125,
  0
},
{ 12208, 8979, 8703, 8682, 5437, 1780, 1303, 1261, 955, 422, 231, 204, 175, 120, 77,
63,
  0
},
{ 13277,10148, 9741, 9698, 6807, 2800, 2047, 1960, 1563, 808, 463, 398, 352, 255, 174,
140,
  0
},
```

도면23e

```
{ 12426, 9284, 8976, 8947, 5753, 2124, 1609, 1555, 1166,  548,  333,  298,  247,  162,  113,
99,
   0
},
{ 13492,10473,10042, 9988, 7177, 3133, 2347, 2244, 1797,  970,  594,  520,  457,  331,
241,  204,
   0
},
{ 12407, 9334, 9013, 8977, 5920, 2276, 1708, 1642, 1284,  642,  392,  348,  299,  211,  153,
132,
   0
},
{ 13579,10631,10182,10121, 7466, 3396, 2542, 2420, 1975, 1112,  691,  604,  542,  408,
300,  256,
   0
},
{ 15415,14163,13591,13335,12111, 9885, 8640, 8070, 7541, 6363, 5413, 4919, 4688, 4157,
3662, 3364,
   0
},
{ 15645,14640,14124,13838,12910,11112, 9969, 9338, 8909, 7923, 7060, 6507, 6315, 5841,
5365, 5003,
   0
},
{ 13803,11130,10691,10636, 7807, 3900, 3070, 2956, 2343, 1300,  866,  781,  667,  462,
337,  292,
   0
},
{ 15381,14112,13568,13348,12083, 9783, 8595, 8105, 7565, 6347, 5408, 4945, 4704, 4158,
3671, 3377,
   0
},
{ 15595,14555,14062,13813,12830,10944, 9820, 9259, 8813, 7769, 6872, 6351, 6149, 5655,
5160, 4798,
   0
},
{ 15855,15124,14719,14487,13812,12440,11517,10975,10628, 9802, 9036, 8493, 8312,
7869,7383, 6992,
   0
},
{ 15484,14324,13800,13578,12460,10325, 9149, 8632, 8118, 6954, 6030, 5548, 5320, 4797,
4296, 3961,
   0
},
{ 15407,14130,13600,13402,12107, 9730, 8527, 8059, 7475, 6205, 5245, 4803, 4563, 4022,
3557, 3260,
   0
},
```

도면23f

```
{ 15475,14322,13830,13621,12482,10384, 9270, 8779, 8266, 7125, 6200, 5730, 5493, 4962,
4469, 4140,
   0
},
{ 15596,14601,14153,13948,12948,11040, 9997, 9517, 9048, 7949, 7054, 6560, 6325, 5785,
5280, 4913,
   0
},
{ 15871,15158,14789,14602,13921,12540,11660,11198,10847,10009, 9224, 8719, 8532,
8078, 7594, 7190,
   0
},
{ 15474,14312,13843,13669,12397,10171, 9109, 8697, 8063, 6764, 5886, 5470, 5181, 4574,
4090, 3790,
   0
},
{ 15720,14858,14443,14249,13373,11680,10700,10226, 9779, 8747, 7881, 7386, 7156,
6611, 6097, 5712,
   0
},
{ 16133,15786,15587,15472,15104,14369,13854,13555,13314,12738,12214,11861,11701,
11314,10905,10579,
   0
},
{  2594,  1645,  1633,  1631,  535,  108,  92,  89,  64,  34,  27,  25,  21,  15,  12,  11,
   0
},
{  7130,  4521,  4444,  4431,  2088,  598,  489,  469,  378,  235,  185,  173,  153,  120,  100,
94,
   0
},
{  1326,  780,  778,  777,  173,  17,  15,  14,  11,  7,  6,  5,  4,  3,  2,  1,
   0
},
{  4974,  2832,  2814,  2812,  945,  102,  78,  76,  56,  26,  16,  15,  13,  9,  6,  5,
   0
},
{  3593,  2051,  2043,  2042,  530,  36,  28,  27,  18,  8,  6,  5,  4,  3,  2,  1,
   0
},
{  5521,  3055,  3028,  3026,  1052,  102,  72,  69,  49,  20,  12,  11,  9,  6,  4,  3,
   0
},
{  4294,  2539,  2520,  2518,  812,  83,  61,  59,  41,  17,  12,  11,  9,  6,  4,  3,
   0
},
```

도면23g

```
{ 4456, 2667, 2653, 2652, 724,  67,  54,  53,  31,  11,  8,  7,  5,  3,  2,  1,
  0
},
{ 6684, 3922, 3872, 3869, 1425,  186,  133,  129,  86,  35,  21,  19,  15,  9,  6,  5,
  0
},
{ 5087, 3024, 2988, 2985, 965,  99,  71,  69,  43,  15,  10,  9,  7,  4,  3,  2,
  0
},
{ 7587, 4403, 4297, 4289, 1806,  279,  181,  171,  118,  49,  30,  27,  22,  15,  11,  10,
  0
},
{ 6242, 4036, 3962, 3957, 1604,  337,  256,  251,  154,  49,  31,  28,  20,  10,  6,  5,
  0
},
{ 3727, 2352, 2346, 2345, 568,  53,  46,  45,  27,  9,  7,  6,  5,  3,  2,  1,
  0
},
{ 6369, 3904, 3876, 3874, 1372,  186,  150,  147,  101,  42,  27,  25,  20,  13,  9,  8,
  0
},
{ 5012, 3060, 3046, 3045, 921,  76,  60,  59,  37,  13,  9,  8,  6,  4,  3,  2,
  0
},
{ 5471, 3590, 3569, 3568, 1118,  176,  153,  151,  83,  26,  19,  18,  13,  7,  5,  4,
  0
},
{ 5866, 3774, 3736, 3733, 1366,  204,  163,  159,  101,  37,  26,  24,  19,  11,  8,  7,
  0
},
{ 8648, 5538, 5437, 5427, 2527,  521,  397,  382,  269,  121,  81,  75,  60,  40,  30,  27,
  0
},
{ 7300, 5124, 5049, 5043, 2284,  682,  582,  574,  341,  124,  90,  86,  59,  29,  20,  18,
  0
},
{ 7183, 4913, 4817, 4808, 2167,  517,  404,  393,  252,  92,  63,  59,  43,  24,  17,  15,
  0
},
{ 4749, 3285, 3273, 3272, 938,  157,  141,  140,  81,  27,  21,  20,  15,  8,  6,  5,
  0
},
{ 6602, 4705, 4676, 4674, 1689,  388,  347,  344,  192,  63,  49,  47,  33,  16,  11,  10,
  0
},
```

도면23h

```
{ 6849, 4648, 4599, 4596, 1868,  367,  304,  299,  187,  68,  49,  46,  34,  19,  14,  13,
  0
},
{ 16383,16382,14098,13672,13671,13670,11058,10499, 8080, 5456, 4230, 3829, 3412,
2876, 2494, 2264,
  0
},
{ 16383,16382,14436,14062,14061,14060,11574,11038, 8597, 5905, 4621, 4204, 3720,
3106, 2674, 2430,
  0
},
{ 16383,16382,14635,14264,14263,14262,11898,11372, 8910, 6216, 4935, 4506, 3962,
3287, 2835, 2578,
  0
},
{ 16383,16382,14851,14380,14379,14378,12610,12032, 9670, 7609, 6543, 6085, 5335,
4590, 4112, 3825,
  0
}
};
```

2396

도면24

```
unsigned short ari_cf_r [4] = {(3<<14)/4,(2<<14)/4,(1<<14)/4, 0};
```