(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 109473114 A
(43)申请公布日 2019.03.15

(21)申请号 201811478584.8

(22)申请日 2014.03.27

(30)优先权数据
61/806,628 2013.03.29 US
61/857,966 2013.07.24 US
61/891,687 2013.10.16 US
14/226,596 2014.03.26 US

(62)分案原申请数据
201480018639.0 2014.03.27

(71)申请人 苹果公司
地址 美国加利福尼亚

(72)发明人 F·M·鲍姆加特

(74)专利代理机构 中国国际贸易促进委员会专利商标事务所 11038
代理人 罗亚男

(51)Int.Cl.
G10L 19/008(2013.01)
G10L 19/16(2013.01)

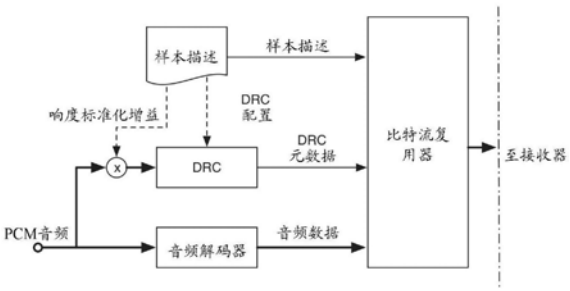
权利要求书1页 说明书33页 附图8页

(54)发明名称

元数据驱动的动态范围控制

(57)摘要

本申请涉及元数据驱动的动态范围控制。描述了一种用于对动态范围控制/压缩(DRC)增益值进行编码并将其应用于一条声音节目内容的系统。具体地,表示一条内容的DRC增益曲线的一组DRC增益值可被分为与该条内容的帧对应的若干个帧。一组字段可与表示一条内容的音频信号一起被包括。附加字段可使用线性插值或样条插值来表示DRC增益值。附加字段可包括:1)用于每个DRC帧的初始增益值,2)DRC曲线中的特定点处的一组斜率值,3)用于每对连续斜率值的一组时间增量值,和/或4)表示斜率值的点之间的DRC增益曲线中的DRC增益值的变化的一个或多个增益增量值。



CN 109473114 A

1.一种对比特流中的表示一条声音节目内容的动态范围控制(DRC)增益值进行编码的方法,包括:

将所述声音节目内容的每个音频声道从一组DRC组分成单个DRC组;

将DRC增益值编码为用于每个DRC组的每个音频帧中的每个DRC子频带的DRC增益数据;以及

将所述DRC增益数据插入到每个DRC组的所述比特流中。

2.根据权利要求1所述的方法,其中在DRC帧中整理所述DRC增益数据,所述DRC帧的尺寸等于所述DRC组中的所述音频帧的尺寸。

3.根据权利要求1所述的方法,其中所述DRC增益数据的用于对所述DRC增益值进行编码的最大采样率为相对于所述声音节目内容的采样间隔的二的整数幂。

4.根据权利要求3所述的方法,其中所述DRC增益数据的最小采样间隔介于1/2ms和1.0ms之间。

5.根据权利要求2所述的方法,还包括:

将编码模式数据值插入到每个DRC组的所述比特流中,其中所述编码模式数据值表示用于对每个对应DRC帧中的所述DRC增益值进行编码的模式。

6.根据权利要求5所述的方法,其中所述编码模式指示DRC组的DRC增益值的数量。

7.根据权利要求5所述的方法,还包括:

将初始增益值插入到每个DRC帧的所述比特流中,其中所述初始增益值指示用于生成对应DRC帧中的后续DRC增益值的起始DRC增益值。

8.根据权利要求7所述的方法,还包括:

将时间增量值插入到成对DRC增益值的所述比特流中,其中所述时间增量值指示每对DRC增益值之间的时间间隔;以及

将增益增量值插入到成对DRC增益值的所述比特流中,其中所述增益增量值指示每对DRC增益值之间的DRC增益值差值。

9.根据权利要求7所述的方法,还包括:

将斜率值插入到每个DRC帧的所述比特流中,其中所述斜率值对应于表示所述DRC增益值的DRC增益曲线的采样点。

10.根据权利要求8所述的方法,其中时间增量值和增益增量值用于由音频回放设备使用插值来将DRC增益应用于对应音频帧。

元数据驱动的动态范围控制

[0001] 本申请是申请日为2014年3月27日申请号为201480018639.0发明名称为“元数据驱动的动态范围控制”的发明专利申请的分案申请。

技术领域

[0002] 本发明的实施例通常涉及一种用于对音频信号进行编码并对其应用动态范围控制/压缩(DRC)的系统和方法。此外,本文所述的系统和方法考虑到开发中的新的编解码器在MPEG-H(3D音频)中的DRC需求。还描述了其他实施例。

背景技术

[0003] 动态范围控制/压缩(DRC)通过(1)使得音频信号中的轻柔部分较响亮;(2)使得音频信号中的响亮部分较轻柔;或(3)同时使得轻柔部分较响亮并使得响亮部分较轻柔来在某种程度上减小音频信号的动态范围。减小的动态范围在若干个情况下可为所期望的,这些情况包括对于可仅再现小的动态范围并同时保持低失真、收听具有分心噪声的环境的音频回放系统的情况,以及在收听者不想打扰其他人的情况。

[0004] 尽管DRC是如今音频编解码器的重要特征,但若干个新近的音频编解码器并不支持DRC。例如,移动图像专家组(MPEG)设定的统一的语音和音频编码(USAC)标准中缺少DRC。高级音频编码(AAC)结合DRC工具,但该DRC工具具有包括有限时间分辨率和混叠失真在内的缺点。

发明内容

[0005] 本发明描述了一种用于对动态范围控制/压缩(DRC)增益值进行编码并将其应用于一条声音节目内容的系统和方法。在一个实施例中,表示一条声音节目内容的DRC增益曲线的一组DRC增益值可被分为与该条声音节目内容的帧对应的若干个帧。附加字段或一组字段可与表示一条声音节目内容的音频信号一起被包括。附加字段可使用线性插值或样条插值来表示DRC增益值。在一个实施例中,附加字段可包括:1)用于每个DRC帧的初始增益值,2)DRC曲线中的特定点处的一组斜率值,3)用于每对连续斜率值的一组时间增量值,以及4)表示与斜率值对应的点之间的DRC增益曲线中的DRC增益值的变化的一个或多个增益增量值。如本文所述,本文的系统和方法提供了一种用于对DRC增益值进行编码并将其应用于一条声音节目内容的有效技术。

[0006] 以上概述不包括本发明的所有方面的详尽列表。可预期的是,本发明包括可由上文概述的各个方面以及在下文的具体实施方式中公开并且在随该专利申请提交的权利要求中特别指出的各种方面的所有合适的组合来实施的所有系统和方法。此类组合具有未在上述发明内容中具体阐述的特定优点。

附图说明

[0007] 本发明的实施例以举例的方式进行说明,而不仅限于各个附图的图示,在附图中

类似的附图标号指示类似的元件。应当指出，本公开中提到“一”或“一个”实施例未必是同一实施例，并且它们表示至少一个实施例。

[0008] 图1示出了根据一个实施例的在音频解码器之后的压缩后处理的框图。

[0009] 图2示出了根据一个实施例的动态范围控制/压缩(DRC)增益表示。

[0010] 图3示出了根据一个实施例的编码器DRC特性。

[0011] 图4示出了根据一个实施例的在发射器处生成的一组示例性DRC元数据。

[0012] 图5示出了根据一个实施例的示例性差值类型。

[0013] 图6示出了根据一个实施例的两个延迟模式。

[0014] 图7示出了根据一个实施例的林奎茨-莱利(Linkwitz-Riley)交叉滤波器的拓扑结构。

[0015] 图8示出了根据一个实施例的利用具有64个子频带的滤波器组的4频带DRC的加权系数的实例。

[0016] 图9示出了根据一个实施例的用于某些对应解码器窗口形状的动态范围控制/压缩(DRC)窗口形状。

[0017] 图10示出了根据一个实施例的应用于单独小块窗口的DRC增益值。

[0018] 图11示出了根据一个实施例的应用于音频信号上的DRC增益值。

具体实施方式

[0019] 现在将参考所附附图来解释本发明的若干个实施例。每当在实施例中描述的部件的形状、相对位置和其它方面未明确限定时，本发明的范围并不仅局限于所示出的部件，所示出的部件仅用于例证的目的。另外，虽然阐述了许多细节，但应当理解，本发明的一些实施例可在没有这些细节的情况下被实施。在其他情况下，未详细示出熟知的电路、结构和技术，以免模糊对本具体实施方式的理解。

[0020] 将动态范围控制/压缩(DRC)元数据结合到比特流/格式中的元数据系统相比于在收听者端处(即，在回放处)确定DRC增益值的系统提供若干个优点。这些优点包括(1)音频信号的回放处的较低复杂性；(2)DRC的复杂性在回放期间出现的问题减少，这就允许实现更复杂的DRC程序；以及(3)收听者端处的音频回放设备可决定是否应用DRC。尽管使用DRC元数据系统提供了若干个优点，但传统的DRC元数据系统诸如由高级电视系统委员会(ATSC)和移动图像专家组(MPEG)所提供的那些传统的DRC元数据系统也提供了若干个缺点。

[0021] 传统的DRC元数据系统(例如，由ATSC和MPEG标准所限定的那些传统DRC元数据系统)支持如表1所示的轻压缩和重压缩。在大多数情况下，DRC增益值更新的速率为每帧一个值。在48kHz的采样率下，这相当于介于21ms和43ms之间的更新间隔。轻压缩模式中的AC-3在48kHz、约5ms的情况下具有快六倍的速率。此外，对于较低音频采样率，这些传统的DRC元数据系统中的DRC增益值以较低速率更新。

[0022]

标准	ATSC:AC-3	MPEG:(HE)AAC
轻压缩	“行模式”	MPEG “动态范围控制”
范围	-24dB…+24dB	-31.75dB…+31.75dB
粒度	0.25dB	0.25dB
速率	每 256 个样本 1 个值	每帧(1024 个或 2048 个样本)1 个值
重压缩	“RF 模式”	DVB “压缩值”
范围	-48dB…+48dB	-48dB…+48dB
粒度	0.5dB	0.5dB

[0023]

速率	每 1536 个样本 1 个值	每帧 1 个值

[0024] 表1:音频标准中的DRC增益元数据的参数

[0025] 实际DRC调谐表明,对于某些音频信号的增益变化应比可利用当前标准所实现的增益变化快得多。

[0026] 当前DRC标准和系统(诸如MPEG-AAC和ATSC)的另一问题源于在应用反相MDCT滤波器组之前将DRC增益应用于频域中的事实。MDCT滤波器组为基于时域混叠消除的转换。如果对连续重叠块应用不同的增益值,则无法实现混叠消除。增益变化可导致听觉失真诸如预回声。这对于响板记录可能很容易示出。

[0027] 在一个实施例中,如果在解码器重建音频信号之后将DRC增益应用于时域中,则可避免MDCT伪像。在频域中,每个长块或每个短块至多可修改增益一次。相比之下,本文所述的时域方法支持所期望的更高时间分辨率。

[0028] 尽管时域方法当前不支持多频带DRC(对MPEG轻压缩可用),但可改进本文所述的实施例以支持多频带DRC。由于插值和DRC增益的应用,本发明提出的方案可稍微增大解码器复杂性。然而,考虑到特别是利用可在高比特率下出现并且可利用高质量回放系统重现的内容来避免不必要的失真,这些缺点显得无关紧要。

[0029] DRC工具

[0030] 概述

[0031] 本文所述的DRC工具基于可应用于时域或子频带域音频信号(诸如HE-AAC解码器的QMF滤波器组的子频带)的统一DRC增益编码。以下描述首先涵盖了时域应用。对于子频带域应用,仅描述了对时域方法的修改。

[0032] 时域应用

[0033] 如图1所示,本节描述了动态压缩工具如何应用于解码之后的时域音频信号。图1示出了在音频解码器之后的压缩后处理的框图。在一个实施例中,DRC工具的解码器部件由元数据来驱动,该元数据有效地表示压缩增益样本和用于插值的参数。在一些实施例中,增益样本可尽可能快地更新以准确地表示下至至少1ms的更新间隔的增益变化。如果增益基本上恒定,则每DRC帧仅使用单个增益样本可足以。为了使比特率最小化,编码器可仅选择足够的DRC增益样本以确保解码之后的音频信号中的重建的DRC增益的足够精度。实际上,在存在较大增益变化的情况下,这可能意味着较小的更新间隔。

[0034] 由于编码器仅提供稀疏采样的增益值，因此解码器可应用插值以实现样本之间的平滑增益转换。经插值的增益的采样率为音频采样率。所使用的插值技术可基于样条。两个相继增益样本之间的一个区段的内插值从该区段的两端处的两个增益样本及其斜率(导数)导出。因此，当从一个区段过渡到下一个区段时，由于该两个区段在过渡点处具有相同斜率，因此一阶导数是连续的。

[0035] 图2示出了基于经量化的DRC增益样本的插值。上面的曲线A示出了以诸如音频采样率的高采样率的DRC输出增益。DRC增益的样本和斜率(即，圆圈和箭头)基于均匀时间网格稀疏地提取。下面的曲线B示出了经量化的增益样本坐标(时间和值)和经量化的斜率，该两者被传输至DRC解码器工具。解码器工具在增益曲线应用于由虚线所示的音频信号之前对其进行插值。

[0036] 在一个实施例中，用于对增益曲线进行采样的最小可能时间间隔为介于0.5ms和1.0ms之间的固定值并且最大可能时间间隔为每DRC帧一个增益样本。

[0037] 除了上述样条模式之外，“简单”模式在没有定时参数和斜率参数的情况下也可用于传输每DRC帧仅一个DRC增益值。该模式最适合于具有基本上恒定DRC增益的帧并且占用最小数量的位。

[0038] 对于DRC工具与音频编解码器协同应用，提供如下参数来调节DRC帧尺寸和时间分辨率，使得编解码器和DRC处理在复杂性和延迟方面可最有效地完成。这些参数为：

[0039] • 以音频采样间隔为单位的DRC帧尺寸

[0040] • 以音频采样间隔为单位的delta_t_min

[0041] • 延迟模式

[0042] 尽管这些参数具有默认值，但编解码器规格可覆写这些默认值。

[0043] 修改DRC特性

[0044] DRC工具支持通过以下若干种方式对经解码的DRC增益进行的修改：

[0045] • 升压因子

[0046] • 压缩因子

[0047] • 自定义DRC特性

[0048] 升压因子为以dB为单位的应用于正增益值以减小放大率的介于0和1之间的值。压缩因子为应用于负增益值以减少衰减的介于0和1之间的值。

[0049] 包括编码器DRC的DRC配置在下文中可称为“样本描述”。例如，前六个静态DRC特性在图3中示出。从概念上来讲，如果此类静态压缩特性无法从DRC算法中明确得出，则该特性可使用1kHz正弦曲线来测量。当峰值在满刻度时，正弦曲线的水平被限定在-3dBFS。图3所示的特性具有下至完全没压缩的不同压缩程度。在最简单的情况下，根据期望压缩效果来选择特性。在还必须控制过载的情况下，例如对于降混，任选地仅可将限制器应用于可能不具有静态压缩效果的编码器中。因此，在仅应用限制器而无DRC的情况下，恒为0dB增益的特性可为有用的。一般来讲，在比特流中传送的DRC增益可为动态压缩或限制或该两者的结果。

[0050] 发射器处的DRC元数据生成的实例在图4中示出。基于根据草案ISO/IEC 14496-12的样本描述来对DRC进行配置。音频信号在其进入DRC之前可为规格化为-31LKFS的响度。DRC元数据可与音频比特流一起被传输。

[0051] 接收器可基于样本描述中所传达的发射器的DRC特性并基于自定义目标DRC特性来修改静态DRC特性。利用所接收的DRC增益值(gainQuant)开始,接收器可应用反相发射器DRC特性,然后应用新的目标DRC特性,如表2所示:

[0052]
```
mapGain(gainQuant) {
 inLevel = inverseCompressorIO(gainQuant);
 outgain = targetCompressorIO(inLevel);
 return outgain;
}
```

[0053] 表2:根据目标DRC特性的DRC增益映射

[0054] 发射器特性1至6的反相可根据表3和表4来计算。请注意,由于增益总是为0dB,因此特性2不具有有用的反相。

[0055]
```
inverseCompressorIO(gainQuant) {
 drcInputLoudnessTarget = -31;
 gainDbMin = -32.0;
 gainDbMax = 32.0;
 if (gainQuant >= 0.0)
  tmp = gainQuant / pow(1 - pow(gainQuant / gainDbMax, expLo), 1 / expLo);
 else
  tmp = gainQuant / pow(1 - pow(gainQuant / gainDbMin, expHi), 1 / expHi);
 inLev = drcInputLoudnessTarget - tmp / ioRatio;
 return inLev;
}
```

[0056] 表3:反相编码器DRC特性1至6的计算

[0057]

参数	Drc 特性					
	1	2	3	4	5	6
io 比率	0.8	0.0	0.2	0.4	0.6	1.0
expLo	6.0	9.0	9.0	9.0	9.0	5.0
expHi	8.0	12.0	12.0	12.0	12.0	6.0

[0058] 表4:DRC特性1至6的参数

[0059] 解码器DRC目标特性不被认为是标准化的。它们可由实现器任选地进行定义以实现自定义的压缩特性。以下章节更详细地解释如何应用增益映射。

[0060] 样本描述可包括总共11个编码器DRC特性。为了与现有系统兼容,除了上述以及图3所示的前6个特性,样本描述还包含表5中所示的能够在ATSC系统中可用的另外五个特性。

[0061]

特性的索引(DRC_characteristic)	简况[5]的名称
7	电影光
8	电影标准
9	音乐光

[0062]

10	音乐标准
11	语音

[0063] 表5:编码器DRC特性7至11的索引

[0064] 样条区段

[0065] 解码器中的DRC增益的插值基于成对的增益样本。每对具有增益坐标(时间和以dB为单位的值)和斜率信息。解码器将选择如图5所示的三个可用类型的插值中的一个插值。在大多数情况下,选择三次插值,该插值由图5中的样条区段A示出。然而,在某些情况下,应用将线性插值和二次插值结合的混合插值来代替,该线性插值和二次插值由图5中的样条区段B和C示出。对于混合插值,在两个增益坐标之间插入节点(示出为图5中的样条区段B和C中的方块)。在该节点的一侧应用线性插值,并且在另一侧应用二次插值。下面对该方法进行充分详细的说明。

[0066] 组帧

[0067] 在DRC帧中整理DRC增益信息。在DRC帧的持续时间内,每个DRC帧包含用于生成DRC增益的DRC数据。对于给定音频项,DRC帧持续时间为恒定不变的并且其为音频采样间隔的倍数。DRC帧不重叠。实际上,只要条件合适,则建议DRC帧尺寸与编解码器的帧尺寸相同以使延迟和复杂性最小化。这可作为默认设置。

[0068] 时间分辨率

[0069] DRC工具使用均匀时间网格来生成DRC增益的稀疏表示。该网格的间距限定了最大可用时间分辨率delta_t_min。delta_t_min的单位在音频采样率下为一个采样间隔。出于复杂性方面的考虑,选择delta_t_min为音频采样间隔的整数倍,其对应持续时间介于[0.5...1.0]ms之间。优选地,delta_t_min为2的整数幂,使得采样率可在音频和DRC之间有效地转换。基于如下公式来计算默认值:

[0070] $delta\_t\_min = 2^{M}$ 其中 $f_s 0.0005s < delta\_t\_min \leq f_s 0.001s$ (1)

[0071] 在上面的公式中,音频采样率f_s以Hz为单位,并且指数M为非负整数。

[0072] 解码器中的预先操作

[0073] DRC工具解码器可在两个延迟模式中的一个延迟模式中操作。低延迟模式立即应用经解码的DRC增益,而默认模式利用一个DRC帧的延迟来应用DRC增益。默认模式支持从当前DRC帧的任何位置到下一DRC帧的任何位置的增益样本插值。低延迟模式要求增益值样本位于DRC帧的末尾处。

[0074] 图6示出了两个延迟模式:低延迟模式A和默认延迟模式B。上面的图示A示出了每个DRC帧具有位于帧末尾处的样条节点,使得针对该帧的整个DRC增益曲线可通过插值立即生成。下面的图示B示出了插值增益曲线利用一个DRC帧的延迟被应用,因为对帧n-1的插值(由圆圈表示)仅能够在接收到帧n的第一节点(由方块表示)之后完成。

[0075] 对于常见的感知编解码器,默认延迟模式B将不需要附加解码器延迟。由于重叠相加操作,已需要该延迟。

[0076] 低延迟模式可适用于不具有固有延迟(诸如由重叠相加所引起的延迟)的解码器。例如,一些无损编解码器就属于这种情况。

[0077] 解码

[0078] 增益坐标和斜率的解码过程包含以下任务序列:

[0079] • 采集DRC配置信息

[0080] • 解析DRC比特流

[0081] • 应用包括哈夫曼解码的代码表以对经量化的值进行解码

[0082] • 撤销差分编码

[0083] DRC配置信息可为样本描述的一部分。DRC配置信息可包括与解码相关的如下参数:

[0084] • 增益序列的数量:nDrcGainSequences

[0085] • 针对每个声道的增益序列的分配。使用相同序列的声道被称为声道组。组的总数为nDrcChannelGroups

[0086] • 组中DRC频带的数量:nDrcBands

[0087] 给出这些参数,可根据表20和表21来解析DRC比特流。在下文中,为了清楚起见,伪码限于一个增益序列。在一般情况下,可添加外循环以处理表6和表9中的每个增益序列。

[0088] 编码值通过应用该表22和表25进行解码。该操作在表6中由伪函数decode_initial_gain()、decode_delta_gain()、decode_time_delta()和decode_slope()来表示。经差分编码的值继而根据表6转换为绝对值。经解码的结果由增益值gDRC[g][b][k]、时间值tDRC[g][b][k]和斜率值sDRC[g][b][k]来表示,其中g为声道组索引,b为频带索引,并且k为样条节点索引。时间值为以delta_t_min为单位的相对于DRC帧起始的整数。与DRC帧的起始一致的音频样本具有tDRC=0的时间值。

[0089]

```
for(g=0; g<nDrcChannelGroups; g++) {
 for(b=0; b<nDrcBands[g]; b++) {
  gDRC[g][b][0] = decode_initial_gain(gain_initial_code[g][b]);
  if (drcGainCodingMode[g][b] == 0) {
   /* "simple" mode */
   tDRC[g][b][0] = drcFrameSize - 1;
   sDRC[g][b][0] = 0.0;
  }
  else
  {
   for (k=1; k<nNodes[g][b]; k++) {
    gDRC[g][b][k]= gDRC[g][b][k-1] + decode_delta_gain(gain_delta_code[g][b][k-1]);
   }

   tDRC[g][b][0] = delta_t_min * decode_time_delta(time_delta_code[g][b][0])- 1;
   for (k=1; k<nNodes[g][b]; k++) {
    tDRC[g][b][k] = tDRC[g][b][k-1] +
```

[0090]

```
                delta_t_min * decode_time_delta(time_delta_code[g][b][k]);
   }

   for (k=0; k<nNodes[g][b]; k++) {
    sDRC[g][b][k] = decode_slope(slope_code[g][b][k]);
   }
  }
 }
}
```

[0091] 表6:dB域中的DRC增益样本坐标和斜率的解码。

[0092] 增益修改和插值

[0093] 如上根据标题"修改DRC特性"所述的,存在在DRC工具解码器中适应DRC特性的若干种方式。将这些调整应用于dB域中的经解码的增益样本。

[0094] 函数toLinear()在表7中被引入,其包括用于从以dB为单位的对数值生成线性增益样本的所有必要步骤(见表7)。该函数包含支持DRC增益值的修改的可选的映射函数mapGain()(见表2),其目的在于实现不同于在编码器中所使用的压缩特性。映射由索引

characteristicIndex控制，该索引在其大于0的情况下将选择自定义解码器DRC特性中的一个自定义解码器DRC特性。否则，将不替换编码器特性。可基于样本描述中所传达的编码器压缩特性来生成经修改的特性。此外，还支持压缩和升压因子以分别缩放负增益和正增益。这些因子则具有1.0的值，除非用户提供范围[0,1]中的值。最后，应用响度标准化增益。

[0095] 在将增益应用于音频信号之前，必须将音频信号转换为线性域和必须内插于增益样本之间的增益值。为了实现较低复杂性，可在插值之前完成dB到线性的转换。因此，插值过程完全是在线性域中完成的。增益修改和到线性域的转换两者均使用表7的伪码来完成。输入变量为dB域中的增益样本和斜率。输出由线性域中的增益样本和斜率组成。对于响度标准化，以dB为单位的响度标准化增益值(loudnessNormalizationGainsDb)可由响度控制工具或其他装置提供至解码器。如果未提供，则使用0.0的默认值。在一个实施例中，标准化增益是以目标响度和内容响应之间的以dB FS为单位的差值来计算的。目标响度为所期望的输出响度水平。内容响度等于如2013年10月在瑞士日内瓦举行的第106届MPEG会议上的ISO/MPEG(106th MPEG meeting Geneva,Switzerland)，“14496-12 PDAM 3-Enhanced Audio(File Format)”中所定义的程序响度或锚响度。如果程序响度和锚响度均未提供，则可将默认值用于内容响度。

[0096]

```
toLinear (gainDb, slopeDb) {
   SLOPE_FACTOR_DB_TO_LINEAR = 0.1151f          /* ln(10) / 20 */
   gainRatio = 1.0;
   if (characteristicIndex > 0) {
      gainRatio = mapGain(gainDb) / gainDb;
   }
   if (gainDb < 0) {
        gainRatio *= compress;
   }
   其他{
        gainRatio *= boost;
   }
   gainLin = pow(2.0, (gainRatio*gainDb+loudnessNormalizationGainDb)/ 6);
   slopeLin = SLOPE_FACTOR_DB_TO_LINEAR * gainRatio * gainLin * slopeDb;
   return (gainLin, slopeLin)
}
```

[0097] 表7:DRC增益样本和相关联的斜率从dB域到线性域的转换

[0098] 增益插值由表8中的伪码来实现。输入变量为：

[0099] • 以目标采样率间隔tGainStep为单位的两个增益样本之间的时间差

[0100] • 以dB为单位的一对相继的增益样本gain0和gain1

[0101] • dB域中的一对对应斜率陡度值slope0和slope1。

[0102] 该函数使用toLinear()来将变量转换为线性域。结果为在位于一对增益样本之间的目标采样率下的平滑增益值序列。目标采样率为经压缩的音频信号的采样率。

[0103]

```
interpolateDrcGain(tGainStep, gain0, gain1, slope0, slope1)
{
   int n;
   float k1, k2, a, b, c, d;

   float slopeLeft;
   float slopeRight;
   float gainLeft;
   float gainRight;

   (gainLeft, slopeLeft) = toLinear (gain0, slope0);
   (gainRight, slopeRight) = toLinear (gain1, slope1);

   slopeLeft = slopeLeft * delta_t_min;
   slopeRight = slopeRight * delta_t_min;

   bool useCubicInterpolation = TRUE;
   int tConnect;
   float tConnectFloat;

   if (abs(slopeLeft) > abs(slopeRight)) {
      tConnectFloat = 2.0 * (gainRight - gainLeft - slopeRight * tGainStep) /
                        (slopeLeft - slopeRight);
      tConnect = (floor) (0.5 + tConnectFloat);
      if ((tConnect >= 0) && (tConnect < tGainStep)) {
         useCubicInterpolation = FALSE;

         result[0] = gainLeft;
         result[tGainStep] = gainRight;

         a = (slopeRight - slopeLeft) / (tConnectFloat + tConnectFloat);
         b = slopeLeft;
         c = gainLeft;
```

[0104]

```
        for (n=1; n<tConnect; n++) {
            float t = (float) n;
            result[n] = (a * t + b ) * t + c;
        }
        a = slopeRight;
        b = gainRight;
        for (   ; n<tGainStep; n++) {
            float t = (float) (n - tGainStep);
            result[n] = a * t + b;
        }
    }
}
else if (abs(slopeLeft) < abs(slopeRight))
{
    tConnectFloat = 2.0 * (gainLeft - gainRight + slopeLeft * tGainStep) /
                        (slopeLeft - slopeRight);
    tConnectFloat = tGainStep - tConnectFloat;
    tConnect = (floor) (0.5 + tConnectFloat);
    if ((tConnect >= 0) && (tConnect < tGainStep)) {
        useCubicInterpolation = FALSE;

        result[0] = gainLeft;
        result[tGainStep] = gainRight;

        a = slopeLeft;
        b = gainLeft;
        for (n=1; n<tConnect; n++) {
            float t = (float) n;
            result[n] = a * t + b;
        }
        a = (slopeRight - slopeLeft) / (2.0 * (tGainStep - tConnectFloat));
        b = - slopeRight;
        c = gainRight;
```

[0105]

```
        for (   ; n<tGainStep; n++) {
            float t = (float) (tGainStep-n);
            result[n] = (a * t + b ) * t + c;
        }
    }
}

if (useCubicInterpolation == TRUE)
{
    float tGainStepInv = 1.0 / (float)tGainStep;
    float tGainStepInv2 = tGainStepInv * tGainStepInv;

    k1 = (gainRight - gainLeft) * tGainStepInv2;
    k2 = slopeRight + slopeLeft;

    a = tGainStepInv * (tGainStepInv * k2 - 2.0 * k1);
    b = 3.0 * k1 - tGainStepInv * (k2 + slopeLeft);
    c = slopeLeft;
    d = gainLeft;

    result[0] = gainLeft;
    result[tGainStep] = gainRight;

    for (n=1; n<tGainStep; n++) {
        float t = (float) n;
        result[n] = (((a * t + b ) * t + c ) * t ) + d;
    }
}
return result;
}
```

[0106] 表8:一个样条区段的DRC增益的插值

[0107] 应用压缩

[0108] 将每个样条区段的内插增益值串联以生成整个DRC帧的完整增益向量gain[g][b][t]。最后,应用如表9所示的增益向量。如果当前声道c属于样本描述中所指定的当前DRC声道组,则函数channelInDrcGroup()则返回TRUE。请注意,对样条区段的调度取决于如表9中所指示的延迟模式(见上文标记“解码器中的预先操作”的章节)。

[0109]

```
for(g=0; g<nDrcChannelGroups; g++) {
 for(b=0; b<nDrcBands[g]; b++) {
  if (delayMode == DELAY_MODE_DEFAULT) {
   for (k=0; k<nNodesPrev[g][b]-1; k++) {
    duration = tDRCprev[g][b][k+1] – tDRCprev[g][b][k];
    splineSegment = interpolateDrcGain(duration, gDRCprev[g][b][k],
                gDRCprev[g][b][k+1], sDRCprev[g][b][k], sDRCprev[g][b][k+1]);
    for (t=0; t<duration; t++)
    {
     gain[g][b][t+tDRCprev[g][b][k]] = splineSegment[t];
    }
   }
   k = nNodesPrev[g][b]-1;
   duration = drcFrameSize + tDRC[g][b][0] – tDRCprev[g][b][k];
   splineSegment = interpolateDrcGain(duration, gDRCprev[g][b][k],
                gDRC[g][b][0], sDRCprev[g][b][k], sDRC[g][b][0]]);
   for (t=0; t<duration; t++)
   {
    gain[g][b][t+tDRCprev[g][b][k]] = splineSegment[t];
   }
  }
  else
  {
   for (k=0; k<nNodes[g][b]-1; k++) {
    duration = tDRC[g][b][k+1] – tDRC[g][b][k];
    splineSegment = interpolateDrcGain(duration, gDRC[g][b][k],
                gDRC[g][b][k+1], sDRC[g][b][k], sDRC[g][b][k+1]);
    for (t=0; t<duration; t++)
    {
     gain[g][b][t+tDRC[g][b][k]] = splineSegment[t];
```

[0110]

```
      }
    }
  }

  /* Apply gain to DRC bands of audio in each channel */

  for (c=0; c<nChannels; c++) {
    if (channelInDrcGroup(c, g)) {
      for (t=0; t<drcFrameSize; t++) {
        audioBandOut[c][b][t] = audioBandIn[c][b][t] * gain[g][b][t];
      }
    }
  }

  if (delayMode == DELAY_MODE_DEFAULT)
  {
    for (k=0; k<nNodes; k++) {
      gDRCprev[g][b][k] = gDRC[g][b][k];
      sDRCprev[g][b][k] = sDRC[g][b][k];
      tDRCprev[g][b][k] = tDRC[g][b][k];
    }
    nNodesPrev[g][b] = nNodes[g][b];
    for (t=0; t<drcFrameSize; t++) {
      gain[g][b][t] = gain[g][b][t + drcFrameSize];
    }
  }
 }
}
for(g=0; g<nDrcChannelGroups; g++) {
 for(c=0; c<nChannels; c++) {
  if (channelInDrcGroup(c, g)) {
    for (t=0; t<drcFrameSize; t++) {
      sum = 0.0;
      for(b=0; b<nDrcBands[g]; b++) {
```

[0111]

```
        sum = sum + audioBandOut[c][b][t];
      }
      audioSampleOut[c][t] = sum;
    }
  }
 }
}
```

[0112] 表9:将样条区段串联成增益向量并且将该DRC增益向量应用于音频声道。

[0113] 表9基于以下假设:

[0114] • splineSegment为包含一个样条区段的增益值的向量。

[0115] • 持续时间为描述以音频采样间隔为单位的样条区段的持续时间的整数。

[0116] • nNodes为当前DRC帧中的增益值的数量。

[0117] • drcFrameSize为DRC帧中的音频采样间隔的数量。

[0118] • 如果delayMode＝＝DELAY_MODE_DEFAULT,则初始化以下变量:gDRCprev[g][b][0]＝0.0,sDRCprev[g][b][0]＝0.0;tDRCprev[g][b][0]＝drcFrameSize;nNodesPrev[g][b]＝1.

[0119] 多频带DRC滤波器组

[0120] 当将DRC增益应用于时域中并且使用多频带DRC时,必须在DRC增益应用于频带之前将时域音频信号分离成子频带。滤波器配置参数可由在MPEG文件格式中所定义的DRCInstructions()来传送。MPEG文件格式可为多个频带和介于频带之间的交叉频率索引提供比特流语法。

[0121] 时域音频信号由具有图7所示的拓扑结构的林奎茨-瑞利(LR)滤波器分离成指定数量的频带。针对2个频带、3个频带和4个频带的林奎茨-瑞利交叉滤波器的拓扑结构在图7中示出。如图7所示,频带索引b随频带频率而增大。交叉频率$f_{C,b}$随索引b(即,$f_{C,b+1}>f_{C,b}$)而增大。在全通滤波器范围内的交叉频率指定具有匹配的相位响应的对应LR低通滤波器。如果存在多于两个的频带,则全通滤波器被添加以补偿不同输出的延迟,使得它们均同相。低通滤波器和高通滤波器实现为二阶区段(双二阶)。

[0122] 如图7所示,每个林奎茨-瑞利(LR)交叉滤波器由形成平坦频率响应的一对互补的低通滤波器和高通滤波器构成。每个LR低通滤波器由两个相同巴特沃斯(Butterworth)(BW)低通滤波器的级联来创建。类似地,每个LR高通滤波器为与BW低通滤波器具有相同阶数和截止频率的两个相同BW高通滤波器的级联。

[0123] 每个BW滤波器和每个全通滤波器实现为具有如下传递函数的二阶区段。

[0124] $$H(z)=\frac{b_0+b_1z^{-1}+b_2z^{-2}}{a_0+a_1z^{-1}+a_2z^{-2}} \qquad (2)$$

[0125] 基于表26中的交叉频率索引,解码器可查找归一化交叉频率$f_{c,Norm}$或滤波器系数参数γ和δ。然后使用用于BW滤波器的表10和用于全通滤波器的表11来计算滤波器系数。以Hz为单位的交叉频率由如下公式来计算f_c:

[0126] $f_c = f_s \cdot f_{c,Norm}$。 (3)

[0127] 在多速率解码器配置诸如双速率HE-AAC的情况下，f_s为最终输出信号的采样率。

[0128]

	BW 低通	BW 高通
归一化截止频率	$\omega_0 = \tan(\pi f_{c,Norm})$	
中间参数	$\delta = \frac{1}{1+\sqrt{2}\omega_0+\omega_0^2}$ $\gamma = \omega_0^2\delta$	
最终 BW 滤波器系数	$a_{LP,0}=1$ $a_{LP,1}=2(\gamma-\delta)$ $a_{LP,2}=2(\gamma+\delta)-1$ $b_{LP,0}=\gamma$ $b_{LP,1}=2\gamma$ $b_{LP,2}=\gamma$	$a_{HP,0}=1$ $a_{HP,1}=2(\gamma-\delta)$ $a_{HP,2}=2(\gamma+\delta)-1$ $b_{HP,0}=\delta$ $b_{HP,1}=-2\delta$ $b_{HP,2}=\delta$

[0129] 表10:巴特沃斯滤波器系数公式

[0130] 图7中的全通滤波器用于生成与LR低通滤波器中的一个LR低通滤波器相同的相位响应(具有图7中匹配的灰度级和匹配的f_c)，使得所有频带的信号在滤波器组的输出处均同相。如表11所示，全通滤波器系数从对应BW低通滤波器的系数导出。

[0131]

$$\begin{aligned} a_{AP,0} &= a_{LP,0} \\ a_{AP,1} &= a_{LP,1} \\ a_{AP,2} &= a_{LP,2} \\ b_{AP,0} &= a_{LP,2} \\ b_{AP,1} &= a_{LP,1} \\ b_{AP,2} &= a_{LP,0} \end{aligned}$$

[0132] 表11:全通滤波器系数公式

[0133] 在将DRC增益应用于单个频带之后，通过添加所有频带来计算最终音频信号。

[0134] 应用于解码器的子频带域的DRC

[0135] 尽管DRC增益在时域中的应用对于AAC是必需的，但其他MPEG编解码器使用子频带域DRC。子频带域DRC的概念意味着解码器的现有子频带信号受DRC增益应用限制。因此，不必要添加用于多频带DRC的时域频带分离并且可能在频域中进行渲染和/或降混之前应用DRC增益。表12包含编解码器和应用DRC增益的域的不完全列表。域可取决于解码器配置而非比特流。例如，如果MPEG-Surround利用普通的AAC解码器来解码，则在时域中应用DRC增益。此外，子频带域可以不是核心编解码器的MDCT域。相反，子频带域通常为QMF域。

[0136]

解码器	时域DRC	子频带DRC
AAC	「	
HE-AAC		「

MPEG-Surround		「
SAOC		「
USAC		「
3D音频		「

[0137] 表12:各种MPEG解码器的DRC增益应用的域

[0138] 为了实现多频带压缩,将压缩器频带映射到解码器子频带组。无需进行附加滤波。DRC交叉频率映射到可用的最靠近的解码器子频带交叉频率。给出用于子频带s的$f_{c,Norm,SB}(s)$归一化子频带交叉频率,$f_{c,Norm}(b)$的经映射的交叉频率为:

[0139]

$$f_{c,Norm,Mapped}(b)=\begin{cases} \text{if } f_{c,Norm,SB}(s)\le f_{c,Norm}(b)\le f_{c,Norm,SB}(s+1): & \\ \quad f_{c,Norm,SB}(s); & \text{if } f_{c,Norm}(b)<0.5\left(f_{c,Norm,SB}(s)+f_{c,Norm,SB}(s+1)\right) \\ f_{c,Norm,SB}(s+1); & \text{else} \end{cases} \tag{4}$$

[0140] 如本文所述,可对DRC增益进行解码。然而可使用表8和表9中所述的相同技术来内插DRC增益,降低插值结果的采样率以匹配子频带信号的采样率。这可由因子L通过对经插值的时域DRC增益进行二次采样或通过使用子频带采样率作为目标来进行直接内插来实现。

[0141] 为了避免DRC频带之间的频谱突变,相邻DRC频带的增益之间可能存在“淡入淡出”。该操作被称为重叠。重叠是由加权系数w来控制的,针对每个子频带存在一个加权系数。加权系数w可根据表14来计算并且确定当前频带的DRC增益的贡献和下一频带的DRC增益的贡献。图8示出了利用具有64个子频带的滤波器组的4频带DRC的加权系数的实例。图8中的频带边缘表示为短划线,频带中心表示为实线,并且权重表示为虚线。

[0142] 在重叠之后,将每个压缩器频带的DRC增益应用于与压缩器频带对应的每个子频带组。将导致滤波器组延迟的小延时D添加至DRC增益以实现与音频信号的适当的时间对准。下采样和延迟操作可由表13中的伪码的第一部分来表示。下文将两个参数的值作为特定于编解码器的值进行论述。表13中伪码的变量和函数的含义在表15中进行说明。描述假设所有子频带中的采样率是相等的。如果不是这种情况,则可针对不同的子频带采样率来调整下采样因子L。

[0143]

```
/* resample DRC gain */
for (g=0; g<nDrcChannelGroups; g++) {
  for (b=0; b<nDrcBands[g]; b++) {
    for (m=0; m<drcFrameSizeSb; m++) {
      gainLr[g][b][m]=gain[g][b][m*L-D];
    }
  }
```

[0144]

```
}

/* Overlap DRC gains in crossover regions
for (g=0; g<nDrcChannelGroups; g++) {
 s=0;
 for (b=0; b<nDrcBands-2; b++) {
  while (fCenterSubband[s] < 0.5*(fCross[g][b] + fCross[g][b+1])) {
   for (m=0; m<drcFrameSizeSb; m++) {
    gainSb[g][s][m] = w[g][s]*gainLr[g][b][m]+(1-w[g][s])*gainLr[g][b+1][m];
   }
   s++;
  }
 }
 for (  ; s<nDecoderSubbands; s++) {
  for (m=0; m<drcFrameSizeSb; m++) {
   gainSb[g][s][m] = w[g][s]*gainLr[g][b][m]+(1-w[g][s])*gainLr[g][b+1][m];
  }
 }
}

/* apply DRC gain in sub-bands */
for (g=0; g<nDrcChannelGroups; g++) {
 for (c=0; c<nChannels; c++) {
  if (channelInDrcGroup(c, g)) {
   for (s=0; s<nDecoderSubbands; s++) {
    for (m=0; m<drcFrameSizeSb; m++) {
     audioSampleSbOut[c][s][m]=gainSb[g][s][m]*audioSampleSbIn[c][s][m];
    }
   }
  }
 }
}
```

[0145] 表13:解码器子频带中的DRC增益下采样、重叠和应用

[0146]

```
/* Overlap DRC gains in crossover regions
olapSize = 0.15;
for (g=0; g<nDrcChannelGroups; g++) {
  bwLeft = 2*fCross[g][0];
  s=0;
  for (b=0; b<nDrcBands-2; b++) {
    olap = olapSize * fCross[g][b];
    fStart = fCross[g][b] - min(0.5*bwLeft, olap);
    bwRight = fCross[g][b+1] – fCross[g][b];
    fStop = fCross[g][b] + min(0.5*bwRight, olap);
    while (fCenterSubband[s] <= fStart) {
      w[g][s] = 1.0;
      s++;
    }
    while (fCenterSubband[s] <= fStop) {
      w[g][s] = (fCross[g][b] + olap - fCenterSubband[s]) / (2 * olap);
      s++;
    }
    while (fCenterSubband[s] <= 0.5 * (fCross[g][b] + fCross[g][b+1])) {
      w[g][s] = 0.0;
      s++;
    }
    bwLeft = bwRight;
  }
  olap = olapSize * fCross[g][b];
  fStart = fCross[g][b] - min(0.5*bwLeft, olap);
  bwRight = 0.5 – fCross[g][b];
  fStop = fCross[g][b] + min(bwRight, olap);
  while (fCenterSubband[s] <= fStart) {
    w[g][s] = 1.0;
    s++;
  }
```

[0147]

```
while (fCenterSubband[s] <= fStop) {
  w[g][s] = (fCross[g][b] + olap - fCenterSubband[s]) / (2 * olap);
  s++;
 }
 while (s < nDecoderSubbands) {
  w[g][s] = 0.0;
  s++;
 }
}
```

[0148] 表14:重叠加权的计算。

[0149]

代码项	含义
gainSb	待应用于解码器子频带的DRC增益
gainLr	低速率(重采样的)DRC增益
fCross	归一化交叉频率
drcFrameSizeSb	一个音频帧中的每子频带的子频带样本数量
nDecoderSubbands	解码器子频带的数量
fCenterSubband	解码器子频带的中心频率
w0,w1	增益重叠的权重
olapSize	重叠区域相对于交叉频率的尺寸
Olap	以归一化频率为单位的重叠区域的尺寸
audioSampleSbIn	动态压缩之前的经解码的子频带音频样本
audioSampleSbOut	动态压缩之后的经解码的子频带音频样本

[0150] 表15:代码项的说明

[0151] 用于传统流式场景的DRC配置

[0152] DRC配置信息可由MPEG文件格式语法来传达。然而,如果使用传统流式格式诸如ADTS来承载不支持MPEG文件格式的MPEG音频流,则可将配置信息嵌入音频流中。这可通过将文件格式的AudioSampleEntry()语法(或其压缩版本)添加至the uni_drc_info()语法来实现。由于样本条目信息仅在相比于帧速率的低速率下才需要,因此可使用指示该信息何时可用的存在标记。扩展语法在表16中给出。

[0153]

uni_drc_info()	
sampleEntryPresent;	1 uimsbf
if (sampleEntryPresent == 1) {	
AudioSampleEntry()	
}	
for (s=0; s<nDrcGainSequences; s++) {	
drc_gain_sequence()	
}	

[0154] 表16:具有样本条目域的扩展uni_drc_info()有效载荷的语法。

[0155] 对于这种情况,DRC信息仅可在解码器接收到样本条目之后被解码。样本条目信息的重复率确定解码延迟。

[0156] 优先性

[0157] 如果比特流包含所提出的DRC元数据和其他DRC元数据诸如MPEG轻压缩或重压缩,则所提出的元数据将具有优先性,除非解码器被指示应用其他DRC元数据。

[0158] 特定于解码器的信息

[0159] 高级音频编码(AAC)

[0160] 用于AAC的DRC元数据扩展

[0161] 对于AAC而言,可使用用于承载填充元素中的所提出的DRC元数据的具有新ID的新扩展有效载荷。ID使用4位代码进行编码并且当前仅定义了7位。将该DRC信息嵌入在新扩展有效载荷中保证了与将忽略新有效载荷的现有解码器的向后兼容性。所提出的新的extension_type在表17中给出。其包含表20中给出的uni_drc_info()。

[0162]

符号	扩展类型的值	目的
UNI_DRC	1111	统一的DRC

[0163] 表17:对AAC的新的extension_type的定义

[0164] AAC的延迟模式

[0165] AAC使用默认延迟模式。

[0166] 用于AAC的DRC帧尺寸和时间分辨率

[0167] DRC帧尺寸具有默认尺寸(即,其具有与AAC帧尺寸相同的持续时间)。

[0168] 如上文标记“时间分辨率”的章节中所指定的,计算在音频采样率下的样本数的delta_t_min的值。为了方便起见,这里基于以下公式和表18来提供具体值:

[0169] $\Delta t_{min} = 2^{M}$ (5)

[0170] 适用指数M可通过查找满足以下条件的音频采样率范围被找到:

[0171] $f_{s,min} \leq f_s < f_{s,max}$ (6)

[0172]

f_s,min[Hz]	f_s,max[Hz]	M
8000	16000	3

16000	32000	4
32000	64000	5
64000	128000	6

[0173] 表18:用于确定AAC的DRC时间分辨率的表

[0174] 给定编解码帧尺寸N_{Codec},在速率delta_t_min下以DRC样本为单位的DRC帧尺寸为:

[0175] $N_{DRC}=N_{Codec}2^{-M}$ (7)

[0176] MPEG-D USAC

[0177] DRC元数据扩展

[0178] 在USAC中,新扩展有效载荷可承载于扩展有效载荷元素UsacExtElement中。为此目的,新扩展元素类型根据

[0179] 表19来定义。应用依赖于默认编解码的DRC设置。

[0180]

符号	扩展类型的值	目的
ID_EXT_ELE_DRC	3	统一的DRC

[0181] 表19:用于USAC的新的usacExtElementType的定义

[0182] MPEG-4 HE-AAC、HE-AACv2、MPEG-D Surround、MPEG-D SAOC

[0183] DRC元数据扩展

[0184] 如上所述,DRC元数据可承载于AAC核心流中。

[0185] MPEG-4HE-AAC、HE-AACv2、MPEG-D Surround和MPEG-D SAOC由核心解码器诸如AAC-LC以及位于该核心解码器的顶部上的一个或多个附加层构成。附加层相比于核心增加了音频带宽或音频声道的数量。对于这些解码器,应在合成滤波器组之前即刻将DRC增益应用于最高层的子频带,但并非在渲染/混合阶段之后进行。

[0186] 子频带中的DRC增益应用

[0187] 对于QMF域中的DRC增益应用,时域DRC增益可由时域采样间隔D延迟并由因子L进行下采样。D和L的值取决于配置,诸如单一速率与双速率HE-AAC。对于所有配置,必须实现DRC增益和音频信号之间的适当时间对准。

[0188] 比特流语法

[0189] DRC比特流在表20和表21中定义。通常,DRC比特流time_domain_drc_info()承载于主机编解码器的扩展有效载荷字段中。

[0190]

```
uni_drc_info()
for (s=0; s<nDrcGainSequences; s++) {
   drc_gain_sequence()
}
```

[0191] 表20:uni_drc_info()有效载荷的语法

[0192]

```
drc_gain_sequence()
for(b=0; b<nDrcBands[g]; b++) {
   drcGainCodingMode[g][b];                    1 uimsbf
   if (drcGainCodingMode[g][b] == 0)
      gain_initial_code[g][b];                 9 uimsbf
   }
   else {
      k=0;
      do {
         slope_code[g][b][k++];                1..10 vlclbf
      }
      while (slope_code[g][b][k-1] != slope_code_end_marker);
      nNodes[g][b] = k;
      for (k=0; k<nNodes[g][b]; k++) {
         time_delta_code[g][b][k];             1..12 vlclbf
      }
      gain_initial_code[g][b];                 9 uimsbf
      for (k=1; k<nNodes[g][b]; k++) {
         gain_delta_code[g][b][k-1];           1..11 vlclbf
      }
   }
}
```

[0193] 表21:drc_gain_sequence()的语法

[0194]

编码	尺寸	gainInitial in [dB]	范围
$\{s, m_1\}$	{1 位，8 位}	$g_{DRC}(0) = (-1)^s m_1 2^{-3}$	-31.875 dB…31.875dB, 0.125dB 步长

[0195] 表22:初始DRC增益值的编码

[0196]

码字尺寸[位]	gainValueDelta 二进制编码	gainDelta[dB]
4	0x000	-2.0
9	0x039	-1.875
11	0x0E2	-1.750
11	0x0E3	-1.625
10	0x070	-1.500

[0197]

10	0x1AC	-1.375
10	0x1AD	-1.250
9	0x0D5	-1.125
7	0x00F	-1.000
7	0x034	-0.875
6	0x036	-0.750
5	0x019	-0.625
5	0x002	-0.500
5	0x00F	-0.375
3	0x001	-0.250
2	0x003	-0.125
3	0x002	0.000
2	0x002	0.125
6	0x018	0.250
6	0x006	0.375
7	0x037	0.500
8	0x01D	0.625
9	0x0D7	0.750
9	0x0D4	0.875
5	0x00E	1.000

[0198] 表23:DRC增益差的编码

[0199]

码字尺寸[bits]	斜率坡度 二进制编码	斜率坡度
7	0x058	-3.0518
9	0x142	-1.2207
8	0x0B2	-0.4883
6	0x02A	-0.1953
6	0x029	-0.0781
6	0x02D	-0.0312
3	0x004	-0.0050
2	0x003	0.000
5	0x017	0.0050
6	0x02B	0.0312
7	0x051	0.0781
10	0x287	0.1953
10	0x286	0.4883
8	0x0A0	1.2207
8	0x0B3	3.0518
1	0x000	n/a(结束标记)

[0200] 表24:斜率坡度的编码

[0201]

码字尺寸[位]	时间差 二进制编码	时间差 tDRC_delta
1	0x000	nNodesMax

[0202]

3	0x004	1
5	0x014+(a-2)	a=[2..5]
6	0x030+(a-6)	a=[6..13]
12	0xE00+(a-14)	a=[14..nNodesMax-1]

[0203] 表25:nNodesMax＝N_{DRC}时的时间差的编码

[0204]

crossover_freq_index	$f_{c,Norm}$	γ	δ
0	2/1024	0.0000373252	0.9913600345
1	3/1024	0.0000836207	0.9870680830
2	4/1024	0.0001480220	0.9827947083
3	5/1024	0.0002302960	0.9785398263
4	6/1024	0.0003302134	0.9743033527
5	2/256	0.0005820761	0.9658852897
6	3/256	0.0012877837	0.9492662926
7	2/128	0.0022515827	0.9329321561
8	3/128	0.0049030350	0.9010958535
9	2/64	0.0084426929	0.8703307793
10	3/64	0.0178631928	0.8118317459
11	2/32	0.0299545822	0.7570763753
12	3/32	0.0604985076	0.6574551915
13	2/16	0.0976310729	0.5690355937
14	3/16	0.1866943331	0.4181633458
15	2/8	0.2928932188	0.2928932188

[0205] 表26:归一化交叉频率和相关联的滤波器系数参数的编码

[0206] 线性插值DRC编码

[0207] 尽管上述涉及基于使用样条插值对增益值进行编码和解码,但在一些实施例中,可使用线性插值来对增益值进行编码和应用。例如,在一个实施例中,可使用如上所述的样条节点来针对一条声音节目内容对DRC值进行编码。在该实施例中,可从比特流中删除每个样条节点之间的斜率值。相反,可在样条节点之间执行线性插值而不是样条插值。以这种方式,DRC增益值的编码可通过避免生成斜率值来简化。

[0208] 基于窗口的重叠相加DRC增益插值

[0209] 在一个实施例中,基于窗口的重叠相加增益插值法可用于对DRC增益值进行解码。在该方法中,以类似于上述的方式对增益值进行编码和接收。然而,每个增益值在解码期间用作窗口的乘法器(例如,窗口系数的向量)。经插值的增益曲线此后可通过使用重叠相加

法来获得。例如，窗口的经插值的DRC增益曲线可为增益值与窗口的乘积。使用窗口的一个原因在于，相比于由在子频带中应用增益值的标准化解码器产生的增益曲线可生成相同的增益曲线。此外，基于窗口的重叠相加增益插值法不生成混叠失真。下文以举例的方式来描述该基于窗口的重叠相加增益插值法的更多深入描述。

[0210] 图9示出了用于某些对应解码器窗口形状(虚线)的DRC窗口形状(实线)。从上到下，图9示出了示例性长窗口、由长到短的过渡窗口、短窗口和由短到长的过渡窗口。DRC窗口可通过解码器合成窗口的平方来计算。DRC窗口可以与对应解码器合成窗口相同的时间被应用。

[0211] 以下公式示出了DRC窗口如何根据具有AAC帧尺寸N的长AAC合成窗口来计算：

[0212] $$w_{DRC,long}(n)=w^2_{AAC,long}(n),\ \text{对于}\ n=[0,2N-1] \qquad (8)$$

[0213] 短窗口和过渡窗口可以类似的方式来计算。以下公式(9)示出了由从比特流导出的DRC增益值加权的连续DRC窗口的重叠相加过程。时间索引和帧索引分别被表示为n和k。时间索引0位于当前合成窗口的起始(当前帧的第一输出样本)处。

[0214] $g(n)=g_{DRC}(k-1)\,w_{DRC}(k-1,N+n)+g_{DRC}(k)\,w_{DRC}(k,n)$ 其中n＝[0,N-1] (9)

[0215] DRC增益继而可x_{AAC}根据以下公式(10)应用于解码器输出信号以生成最终的经压缩的音频输出x_{DRC}。DRC增益并不被应用于MDCT域。

[0216] $x_{DRC}(n)=g(n)\,x_{AAC}(n)$，对于n＝[0,N-1] (10)

[0217] 当在MPEG中使用轻压缩时，多频带DRC元数据可用于将独立DRC增益值应用于单个短块或分组短块。相比于标记“多频带”，可对每个DRC增益进行编码，使其应用于短块的整个MDCT频域。因此，每个DRC增益作为单频带DRC进行操作。如果是这种情况，DRC操作可如上所述类似地在时域中完成。

[0218] 例如，如图10的上面图所示，如果针对8个短块给出5个DRC增益值，则对应DRC窗口被示出为实线。下图示出使用具有相同形状和g_0到g_4的对应DRC增益值的8个短DRC窗口。DRC窗口可从使用具有参数N′＝N/8的公式(8)的短窗口形状导出。通过以参数N′代替N来对应地应用公式(9)和(10)。

[0219] 一般来讲，比特流语法允许独立选择用于单个帧的单频带或多频带DRC。利用上述时域DRC具体实施，无论何时存在真实的多频带DRC增益信息(“真实的多频带”意味着针对不同的子频带存在不相等的DRC增益值)，解码器都将切换到MDCT域DRC处理。

[0220] 该提案包括经修改的MPEG-AAC DRC具体实施，该具体实施通过单频带DRC的向后兼容的方式避免了混叠失真。尽管上述涉及MPEG-AAC DRC，但在其他实施例中，可使用包括频域DRC增益值的任何类型的比特流音频。

[0221] 在上述实施例中，解码器被修改以将DRC应用于时域。在另一个实施例中，可将附加字段添加至比特流以提高DRC增益值在时域中应用于音频信号的可变性。DRC增益值的新字段可在比特流语法中的不同位置处定义。对于MPEG标准，一个选项是表27中所示的填充元素中所携带的附加扩展有效载荷的定义。在该实施例中，可将程序内容的音频声道分成DRC组，其中每个组具有独立的一组DRC信息，即将单独的独立DRC应用于每组声道。音频声道可仅属于一个DRC组或不属于DRC组。可将分组信息添加至样本描述，其在轨道的起始出现一次。在该实施例中，DRC组的数量被称为nDrcChannelGroups。

[0222]

元数据序列

For(g=0; g<nDrcChannelGroups-1; g++) {
k=0;
nDrcGainInfoBlocks[g]; 2 uimsbf
for (b=0; b<nDrcGainInfoBlocks[g]; b++) {
drcGainCodingMode[g][b]; 2 uimsbf
if (b==0) {
gainInitial[g]; 9 uimsbf
for (n=1; n<nDrcGainValues[g][b]; n++) {
gainDelta[g][k++]; 1..11 vlclbf
}
}
else {
for (n=0; n<nDrcGainValues[g][b]; n++) {
gainDelta[g][k++]; 1..11 vlcbf
}
}
}
}

[0223] 表27:时域DRC扩展有效载荷

[0224] 在观察实际具体实施中随时间变化的增益时,可看出增益有时可变化非常缓慢,而在音频信号表现出攻击时增益可表现出显著变化。用于对DRC增益值进行编码的必要比特率通过支持用于每个所谓的drcGainInfoBlock的单个可选的时间分辨率而降低。音频帧均匀分成多表28中所示的最多至8个这些信息块并且每个信息块可包含最多至16个增益值。

[0225] 与增益值的较大时间分辨率相关联的比特率增加进一步通过使用利用增益变化的熵编码进行的自适应方案来减轻。DRC增益值可使用表27中定义的语法在每个音频帧中传输。

[0226]

码字	nDrcGainInfoBlocks	注释

[0227]

0x0	1	每帧 1 个块
0x1	2	每帧 2 个块
0x2	4	每帧 4 个块
0x3	8	每帧 8 个块

[0228] 表28:nDrcGainInfoBlocks的查找表

[0229] 条目drcGainCodingMode确定表29中所给出的信息块的增益值的数量。每帧可存在至少一个增益值以支持随机插入。第一增益值根据表30来编码。其余增益值使用表31或表32(根据所选择的drcGainCodingMode)进行差分编码。

[0230]

drcGainCodingMode (码字)	nDrcGainValues	drcDiffGainFactor	注释
0x0	1	n/a	每帧 1 个值
0x1	4	1	每帧 4 个值
0x2	16	1	每帧 16 个值
0x3	16	4	每帧 16 个值

[0231] 表29:drcGainCodingMode的查找表

[0232]

编码	尺寸	gainInitial in [dB]	范围
$\{s, m_1\}$	{1 位，8 位}	$g_{DRC}(0) = (-1)^s m_1 2^{-3}$	-31.875 dB...31.875dB, 0.125dB 步长

[0233] 表30:gainInitial的表示

[0234]

gainValueDelta (二进制代码)	gainDelta[dB] (分辨率)
00011000010	-1.5
000110001	-1.0
0001101	-0.75
000111	-0.5
0000	-0.375
001	-0.25
10	-0.125
11	0.0
01	0.125
00010	0.25
00011001	0.375
0001100000	0.5
00011000011	1.0

[0235] 表31:drcGainCOdingMOde＝＝1时的DRC增益差的编码

[0236]

gainValueDelta (二进制代码)	gainDelta[dB] (分辨率)
1100001110	-4.0
110000110	-3.0
1100000	-2.0
110001	-1.5
11010	-1.0
11011	-0.75
1111	-0.5
100	-0.25
101	-0.125
0	0.0
1110	0.125
11001	0.25
110000100	0.5
110000101	0.75
1100001111	1.25

[0237] 表32:drcGainCodingMode∈[2,3]时的DRC增益差的编码

[0238]

```
float gDRC[][];
for(g=0; g<nDrcChannelGroups-1; g++) {
  int k=0;
  for (b=0; b<nDrcGainInfoBlocks[g]; b++) {
    if (b==0) {
      gDRC[g][k++] = gainInitial[g];
      for (n=1; n<nDrcGainValues[g][b]; n++) {
        gDRC[g][k] = gDRC[g][k-1] + drcDiffGainFactor[g][b] * gainDelta[g][k-1];
        k++;
      }
    }
    else {
      for (n=0; n<nDrcGainValues[g][b]; n++) {
        gDRC[g][k] = gDRC[g][k-1] + drcDiffGainFactor[g][b] * gainDelta[g][k-1];
        k++;
      }
    }
  }
  float gDRCPrev[g] = gDRC[g][k-1];
}
```

[0239] 表33:对数DRC增益值[dB]的解码

[0240] 差值的非均匀分辨率是由心理声学引起的,诸如观察到增益变化上的偏差越不易听到,增益变化越大。反之亦然,如果增益几乎恒定不变(并且音频包络也恒定不变),则增益变化上的偏差更容易听到。非对称范围适用于对音频信号中的突发攻击进行快速反应DRC增益衰减。增益增加通常较慢。

[0241] 典型的音频解码器使用重叠相加法利用与后续块50%的重叠来重建音频信号。每个块由在任一端渐缩的窗口来加权。例如,MPEG-AAC的典型帧尺寸为1024个样本。对于每个新帧,解码器重建2048个样本,其中将前1024个样本添加到前一个块的后1024个样本并且结果为解码器输出。在重建块的后半部分期间均匀调度具有帧k的信息块。每个信息块内的增益值均匀分布在信息块的持续时间内。该方案确保了所有必要DRC增益值在对起始和末尾进行解码时并且对于插值均为可用的。

[0242] 图11示出了其中比特流的帧n包含用于合成窗口的后半部分的DRC增益。帧n具有4个信息块,这些信息块分别具有1个、8个、2个和4个DRC增益值。DRC增益值的定时基于增益值在每个信息块内的均匀分布来计算。随后,使用线性插值来生成每个时域音频样本的增益值。

[0243]

```
for(g=0; g<nDrcChannelGroups-1; g++) {
 int k=0;
 float tGainPrev = 0.0;
 float deltaPrev = 0.0;
 float samplesPerIblock = nFrame/nDrcGainInfoBlocks[g];
 for (b=0; b<nDrcGainInfoBlocks[g]; b++) {
  float delta = samplesPerIblock / nDrcGainValues[g][b];
  for (n=0; n<nDrcGainValues[g][b]; n++) {
   float tGain[g][k] = tGainPrev + 0.5 * (delta + deltaPrev);
   tGainPrev = tGain[g][k];
   deltaPrev = delta;
   k++;
  }
 }
}
```

[0244] 表34:计算DRC增益值的时间位置

[0245] 增益值定时的计算在表34中给出。结果tGain[g][k]指示以采样间隔为单位的始于当前输出帧的第一样本处的0.0的样本位置。帧尺寸以N_{frame}样本来表示。

[0246] 给出增益值和它们的定时,针对当前输出帧的所有样本的平滑增益曲线可通过如表35所示的线性增益值的线性插值来构建。gDRCprev为前一帧的最后DRC增益值。在该实施例中,需要下一帧的第一增益值来内插帧的增益值以用于输出。由于重叠相加过程,该增益值是可用的,而无需在比特流之前进行额外读取。引入函数toLinear()以包括用于从以dB

为单位的对数值生成线性增益值的所有必要步骤。

[0247]

```
for(g=0; g<nDrcChannelGroups-1; g++) {
  int k=0;
  float tLeft = tGainPrev[g] - nFrame;
  float tRight = tGain[g][0];
  float gainLeft = toLinear(gDRCPrev[g]);
  float gainRight = toLinear(gDRC[g][0]);
  float factor = (gainRight-gainLeft)/(tRight - tLeft);
  for (t=0; t<nFrame; t++) {
    while (tTarget > tRight) {
      k++;
      tLeft = tRight;
      tRight = tGain[g][k];
      gainLeft = gainRight;
      gainRight = toLinear(gDRC[g][k]);
      factor = (gainRight-gainLeft)/(tRight - tLeft);
    }
    gain[g][t] = gainLeft + factor * (t – tLeft);
  }
}
```

[0248] 表35:DRC增益值的插值

[0249] 最后,如表36所示,应用经插值的DRC增益。

[0250]

```
for(g=0; g<nDrcChannelGroups-1; g++)
  for (ch=0; ch<nChannels; ch++) {
    if (channelInDrcGroup(ch)) {
      for (t=0; t<nFrame; t++) {
        audioSampleOut[ch][t] = audioSampleIn[ch][t] * gain[g][t];
      }
    }
  }
}
```

[0251] 表26:DRC增益值的应用

[0252] 上述该实施例包括针对音频标准诸如MPEG-Audio的改进的DRC元数据编码和处理。已解决当前标准的缺点,诸如生成混叠失真以及DRC元数据的时间分辨率不足。

[0253] 如上所述,多种技术可用于对DRC增益值进行编码并将其应用于一条声音节目内

容。在一些实施例中，一种用于将频域动态范围控制(DRC)增益值应用于时域中的音频信号的方法，包括：接收比特流，其中该比特流包括经编码的音频信号和频域DRC增益值；由回放设备中的解码器对经编码的音频信号进行解码以产生时域中的经解码的音频信号；由解码器来确定用于将频域DRC增益值应用于时域中的经解码的音频信号的DRC窗口权重；基于频域DRC增益值和DRC窗口权重来确定时域DRC增益值；以及将时域DRC增益值应用于时域中的经解码的音频信号的对应帧。

[0254] 在一个实施例中，DRC窗口权重基于解码器的合成窗口来确定。在一个实施例中，DRC窗口权重利用与解码器的合成窗口相同的定时作为解码器合成窗口的平方来计算。在一个实施例中，DRC窗口权重基于解码器的合成窗口和编码器的窗口的乘积来确定。在一个实施例中，经解码的音频信号的当前帧的时域DRC增益值基于利用所应用的对应DRC窗口权重的当前帧的频域DRC增益值以及利用所应用的对应DRC窗口权重的前一帧的频域DRC增益值来确定。在一个实施例中，应用时域DRC增益值以产生时域中的DRC音频信号是基于时域DRC增益值和经解码的音频信号的对应时间分割的乘积而进行的。在一个实施例中，时域DRC增益值中的一个或多个时域DRC增益值应用于经解码的音频信号的整个DRC窗口。在一个实施例中，经编码的音频信号为移动图像专家组-高级音频编码(MPEG-AAC)DRC音频信号。在一个实施例中，经编码音频信号为高级电视系统委员会(ATSC)DRC音频信号。

[0255] 在一个实施例中，一种对比特流中的表示一条声音节目内容的动态范围控制(DRC)增益值进行编码的方法，包括：将声音节目内容的每个音频声道从一组DRC组分成单个DRC组；以及将DRC增益元数据插入到每个DRC组的比特流中，其中每个DRC组的DRC增益元数据用于将对应DRC增益值可变地应用于DRC组中的每个帧。在一个实施例中，每个DRC组的DRC增益元数据包括：表示针对初始DRC增益值的所选择的编码模式的第一数据值；表示初始DRC增益值的第二数据值；以及表示应用于初始DRC增益值的差值以生成DRC组的每个帧的DRC增益值的第三数据值。在一个实施例中，第一数据值表示基于初始DRC增益值应用于DRC组的每个帧的增益值的数量。在一个实施例中，由第一数据值所表示的所选择编码模式选自一组预定义编码模式。在一个实施例中，DRC增益值使用插值而被应用。在一个实施例中，插值为线性域中的线性插值。在一个实施例中，将多个声道分配给单个DRC组。在一个实施例中，非均匀时间分辨率基于由编码器DRC所生成的增益变化用于DRC增益值的更新率以使比特流的比特率最小化。在一个实施例中，表示初始增益值的第一数据值使用基于心理声学的非均匀量化标度来编码以使比特流的比特率最小化。在一个实施例中，表示初始增益值的第一数据值使用可变长度代码来编码以使比特流的比特率最小化。在一个实施例中，对表示应用于DRC组的每个帧的初始DRC增益值的差值第三数据值进行编码以使比特流的比特率最小化。在一个实施例中，对表示应用于初始DRC增益值的差值的第三数据值进行可变长度代码编码以使比特流的比特率最小化。

[0256] 如上所述，本发明的一个实施例可为具有存储在其上的指令的机器可读介质诸如一个或多个固态存储器设备，该指令对一个或多个数据处理部件(本文中一般称为“处理器”或“计算机系统”)机型编程以执行上述操作中的一些操作。在其他实施例中，可通过包含硬连线逻辑部件的特定硬件部件来执行这些操作中的一些操作。可替代地，可通过经编程的数据处理部件和固定硬连线电路部件的任何组合来执行那些操作。

[0257] 虽然已描述并且在附图中示出了某些实施例，但应当理解，此类实施例仅用于说

明广义的发明而非对其进行限制，并且本发明并不限于所示和所述的特定构造和布置，因为对于本领域的普通技术人员而言可想到各种其它修改。因此，要将描述视为示例性的而非限制性的。

[0258] 相关问题

[0259] 本专利申请要求于2013年3月29日提交的美国临时专利申请61/806,628；于2013年7月24日提交的美国临时专利申请61/857,966；以及于2013年10月16日提交的美国临时专利申请61/891,687的较早申请日期的权益。

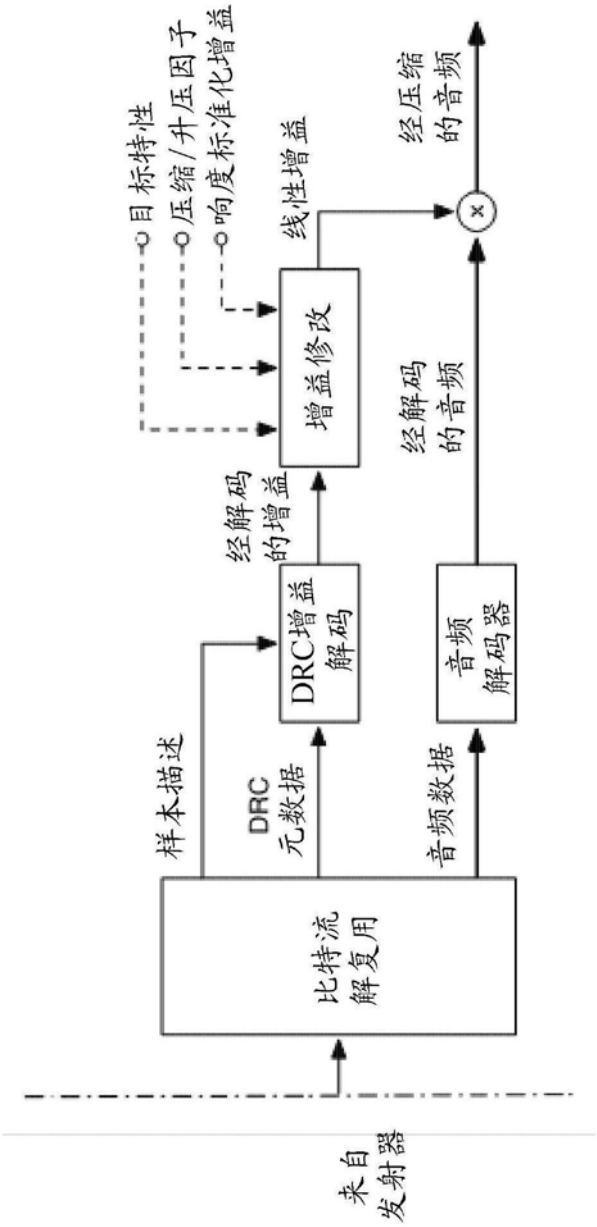

来自
发射器
比特流
解复用
样本描述
DRC
元数据
音频数据
DRC增益
解码
音频
解码器
经解码
的增益
经解码
的音频
增益修改
线性增益
经压缩
的音频
目标特性
压缩/升压因子
响度标准化增益
×

图1

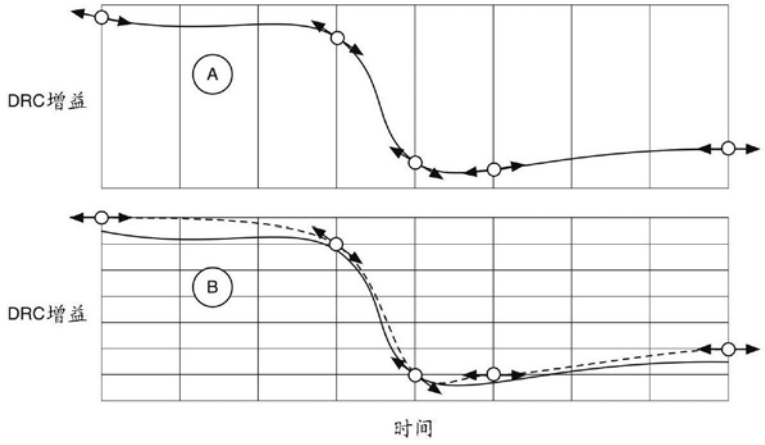

A
DRC增益
B
DRC增益
时间

图2

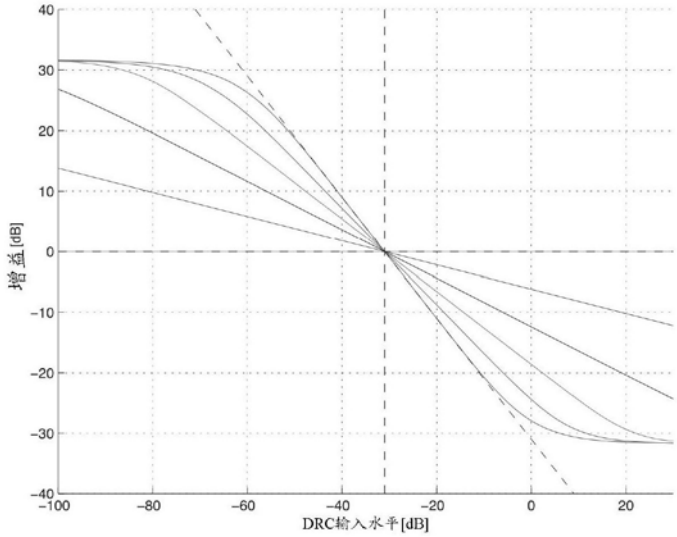

40
30
20
10
0
-10
-20
-30
-40
增益[dB]
-100
-80
-60
-40
-20
0
20
DRC输入水平[dB]

图3

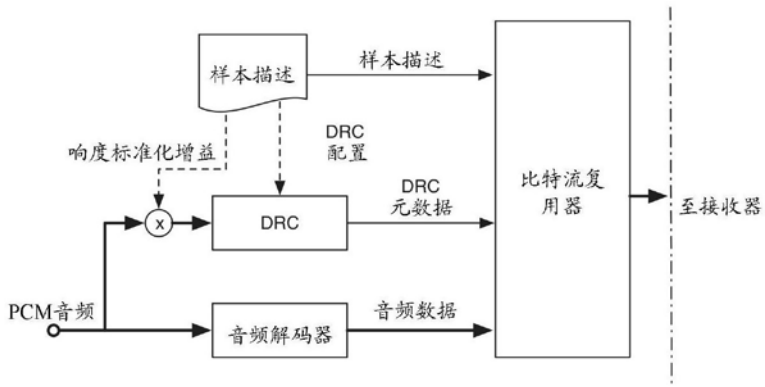

样本描述
样本描述
响度标准化增益
DRC
配置
DRC
元数据
比特流复用器
至接收器
X
DRC
PCM音频
音频解码器
音频数据

图4

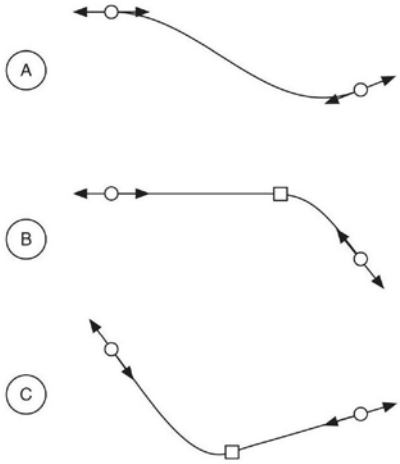


图5

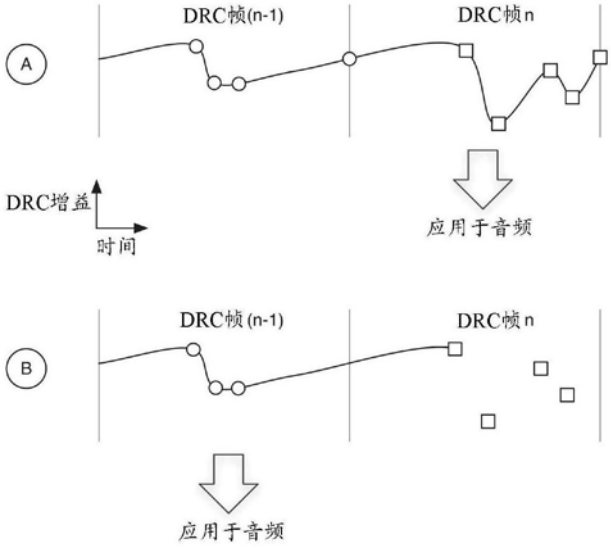


图6

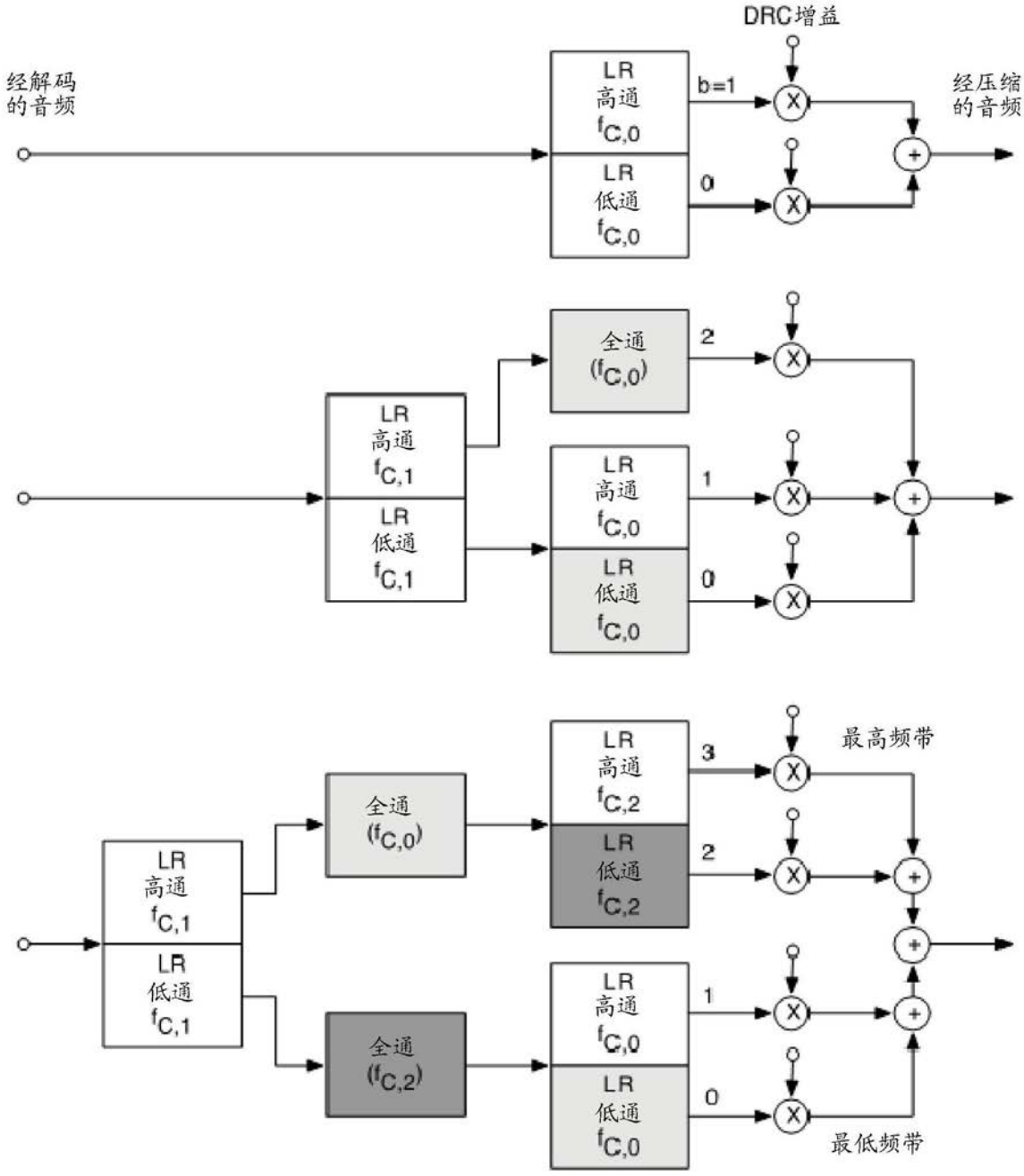

DRC增益
经解码的音频
LR 高通 $f_{C,0}$
LR 低通 $f_{C,0}$
b=1
0
经压缩的音频
LR 高通 $f_{C,1}$
LR 低通 $f_{C,1}$
全通 ($f_{C,0}$)
2
1
0
全通 ($f_{C,2}$)
3
最高频带
最低频带

图7

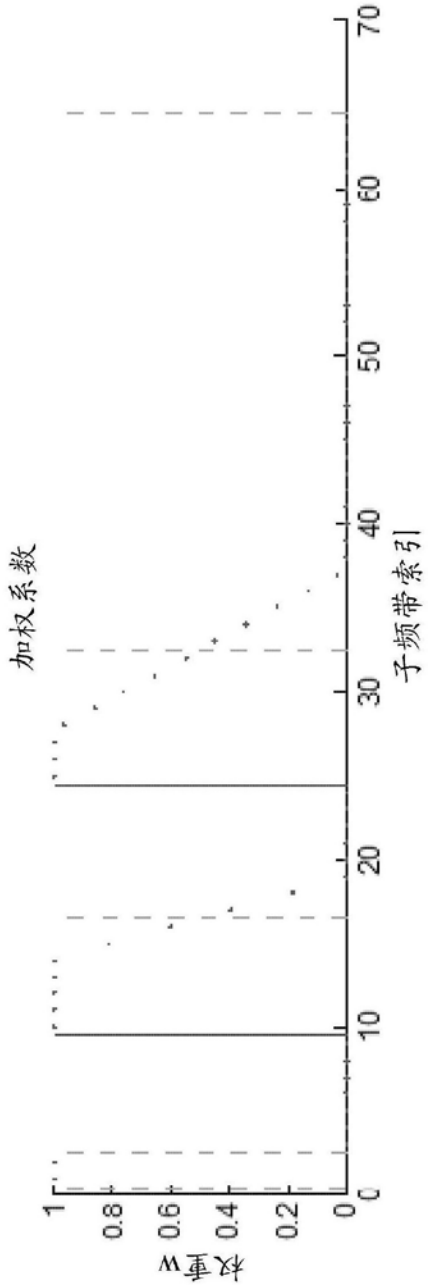


图8

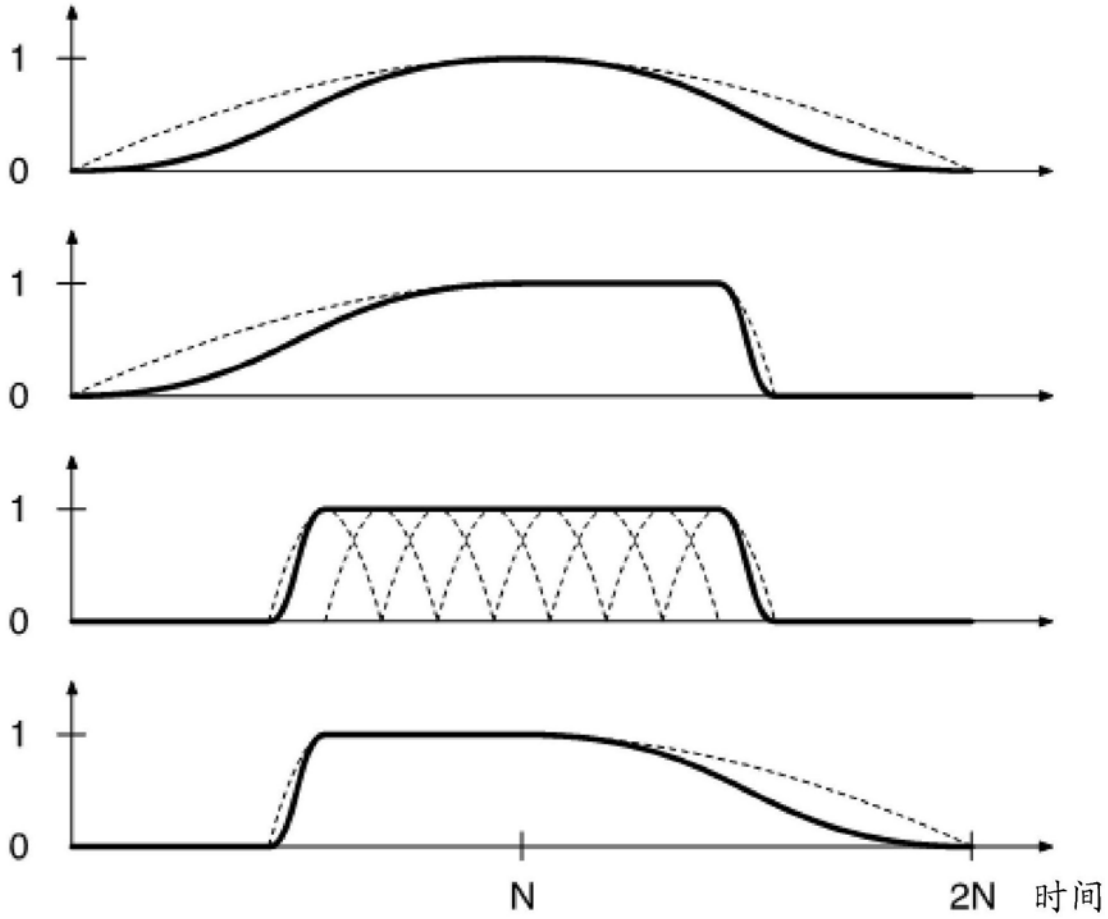


图9

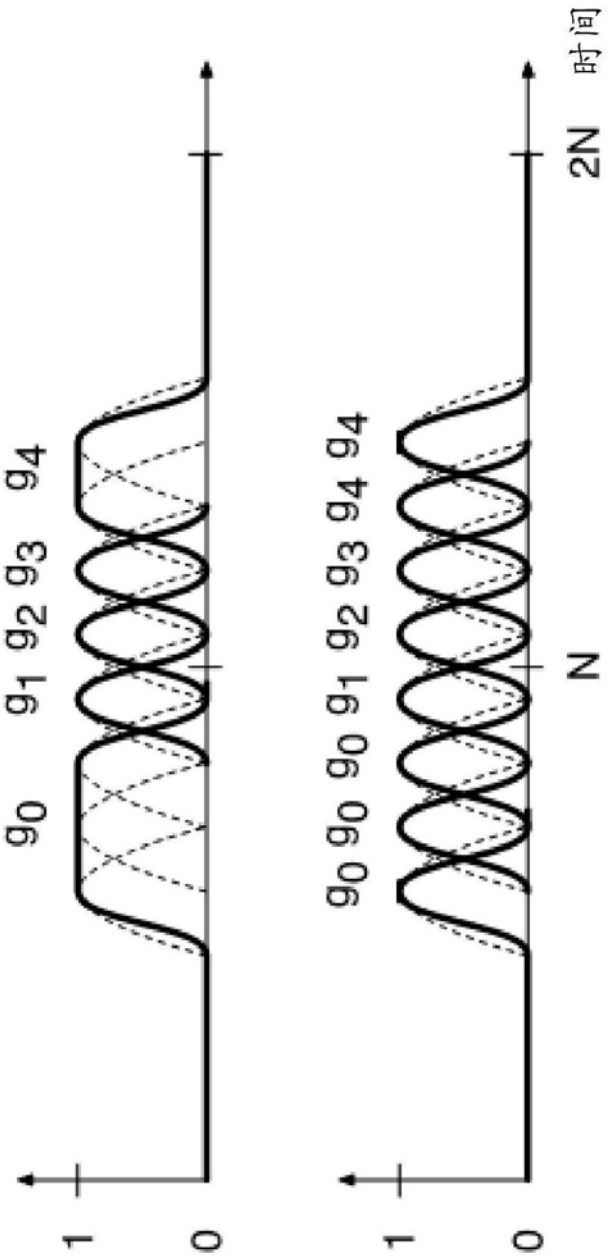


图10

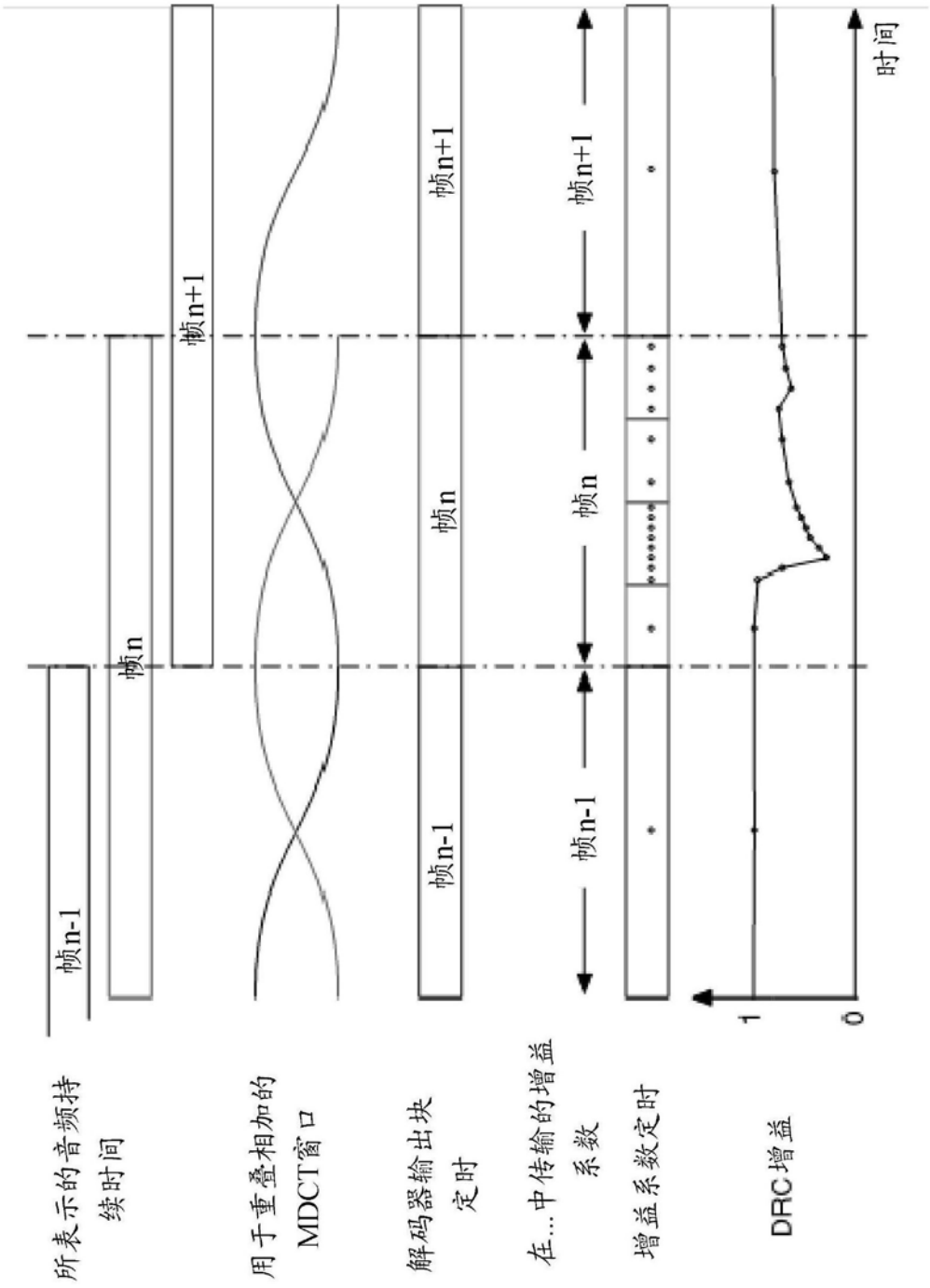

帧n-1
帧n
帧n+1
所表示的音频持续时间
用于重叠相加的MDCT窗口
解码器输出块定时
在...中传输的增益系数
增益系数定时
DRC增益
1
0
时间

图11