

[19] 中华人民共和国国家知识产权局

[51] Int. Cl.

G10L 19/00 (2006.01)

H04N 7/64 (2006.01)



[12] 发明专利申请公布说明书

[21] 申请号 200680029417.4

[43] 公开日 2008年8月13日

[11] 公开号 CN 101243489A

[22] 申请日 2006.7.10

[21] 申请号 200680029417.4

[30] 优先权

[32] 2005.7.11 [33] US [31] 60/697,551

[32] 2005.7.16 [33] KR [31] PCT/KR2005/002290

[32] 2005.7.16 [33] KR [31] PCT/KR2005/002291

[32] 2005.7.16 [33] KR [31] PCT/KR2005/002292

[32] 2005.7.18 [33] KR [31] PCT/KR2005/002306

[32] 2005.7.18 [33] KR [31] PCT/KR2005/002307

[32] 2005.7.18 [33] KR [31] PCT/KR2005/002308

[32] 2005.7.19 [33] US [31] 60/700,570

[86] 国际申请 PCT/KR2006/002677 2006.7.10

[87] 国际公布 WO2007/007999 英 2007.1.18

[85] 进入国家阶段日期 2008.2.13

[71] 申请人 LG 电子株式会社

地址 韩国首尔

[72] 发明人 T·利伯成

[74] 专利代理机构 上海专利商标事务所有限公司

代理人 侯颖嫫

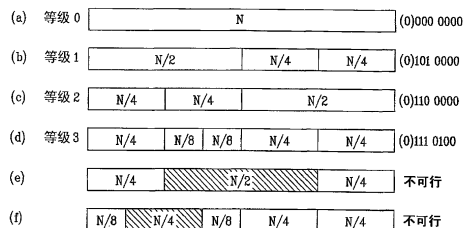
权利要求书 3 页 说明书 29 页 附图 6 页

[54] 发明名称

编码和解码音频信号的装置和方法

[57] 摘要

在一个实施例中，音频信号的一个帧中的至少一个声道被细分成多个块以使这些块中的至少两个具有不同的长度，并且生成指示该声道被细分成若干块的信息。



1. 一种处理音频信号的方法，包括：

将所述音频信号的一个帧中的至少一个声道细分成多个块，所述多个块中的至少两个具有不同的长度；以及

生成指示所述声道被细分成多个块的信息。

2. 如权利要求 1 所述的方法，其特征在于，所述细分步骤根据细分分级结构来细分所述声道，所述细分分级结构具有一个以上的等级，并且每个等级关联于一个不同的块长度。

3. 如权利要求 2 所述的方法，其特征在于，所述细分分级结构的上级关联于是与下级相关联的块长度的两倍的块长度。

4. 如权利要求 2 所述的方法，其特征在于，所述细分步骤细分所述上级的块以在所述下级获得两个块。

5. 如权利要求 2 所述的方法，其特征在于，如果声道具有长度 N ，则所述细分步骤将所述声道细分成多个块以使每个块具有 $N/2$ 、 $N/4$ 、 $N/8$ 、 $N/16$ 、 $N/32$ 中的一个的长度。

6. 如权利要求 2 所述的方法，其特征在于，所述细分步骤将声道细分成多个块以使每个块的长度等于以下之一：

$$N/(m^i) \quad , \quad i=1, 2, \dots, p,$$

其中 N 是该声道的长度， m 是大于或等于 2 的整数，而 p 表示所述细分分级结构中的级数。

7. 如权利要求 6 所述的方法，其特征在于， $m=2$ 并且 $p=5$ 。

8. 如权利要求 2 所述的方法，其特征在于，所述生成步骤生成信息以使所述信息的长度取决于所述细分分级结构中的级数。

9. 如权利要求 2 所述的方法，其特征在于，所述生成步骤生成信息以使所述信息包括若干信息比特，并且所述信息比特指示所述声道被细分成若干块。

10. 如权利要求 9 所述的方法，其特征在于，每个信息比特关联于所述细分分级结构中的一级并与相关联的等级上的一个块相关联，并且每个信息比特指示关联的块是否被细分。

11. 如权利要求 10 所述的方法，其特征在于，如果所述信息比特具有值 1，

则相关联的块是被细分的，并且如果所述信息比特具有值 0，则相关联的块不被细分。

12. 如权利要求 1 所述的方法，其特征在于，还包括：
发送所述信息。

13. 如权利要求 1 所述的方法，其特征在于，还包括：
基于之前的数据采样预测所述声道中的当前数据采样，在所述预测步骤中使用的之前的数据采样的数目是预测阶数；以及

基于所预测出的数据采样获得所述当前数据采样的残差。

14. 如权利要求 13 所述的方法，其特征在于，当之前的数据采样可用时，所述预测步骤渐进地增大预测阶数至合需的预测阶数。

15. 如权利要求 14 所述的方法，其特征在于，
不具有之前的数据采样的开头的数据采样的预测阶数为零，并且所述预测步骤生成零作为所预测出的开头数据采样；并且作为所预测出的开头的数据采样为零的结果，所述获得步骤实质上不改变所述开头的数据采样。

16. 如权利要求 14 所述的方法，其特征在于，还包括：
基于最大允许预测阶数和块长度确定每个块的合需预测阶数。

17. 如权利要求 13 所述的方法，其特征在于，还包括：
基于最大允许预测阶数和块长度确定每个块的最优预测阶数。

18. 如权利要求 17 所述的方法，其特征在于，所述确定步骤包括：
基于所述最大允许预测阶数确定全局预测阶数；
基于所述块长度确定本地预测阶数；以及
将所述全局预测阶数和所述本地预测阶数中最小的一个选为所述最优预测阶数。

19. 如权利要求 13 所述的方法，其特征在于，如果所述帧是以无需用到之前的帧来对其解码的方式编码的随机存取帧，则

当仅来自所述随机存取帧的之前数据采样可用时，所述预测步骤渐进地增大预测阶数至合需的预测阶数。

20. 一种对音频信号进行编码的方法，包括：

将所述音频信号的一个帧中的至少一个声道细分成多个块，所述多个块中的至少两个具有不同的长度；

生成指示所述声道被细分成若干块的信息；以及

对所述多个块进行编码以生成包含所述信息的压缩比特流。

21. 一种对音频信号进行解码的方法，包括：

接收具有至少一个声道的音频数据帧。所述声道被细分成多个块，所述多个块中的至少两个具有不同的长度；

从所述音频信号获得指示所述声道被细分成若干块的信息；以及
基于所获得的信息对所述声道进行解码。

22. 一种对音频信号进行编码的装置，包括：

编码器，所述编码器被配置成将所述音频信号的一个帧中的至少一个声道细分成多个块，所述多个块中的至少两个具有不同的长度，并且所述编码器被配置成生成指示所述声道被细分成若干块的信息并对所述多个块进行编码以生成含所述信息的压缩比特流。

23. 一种对音频信号进行解码的装置，包括：

解码器，所述解码器被配置成接收具有至少一个声道的音频数据帧，所述声道被细分成多个块，所述多个块中的至少两个具有不同长度，并且所述解码器被配置成从所述音频信号获得所述声道被细分成若干块的信息并基于所获得的信息对所述声道进行解码。

编码和解码音频信号的装置和方法

发明背景

本发明涉及一种处理音频信号的方法，尤其涉及一种编码和解码音频信号的方法和装置。

过去曾经以不同方法实现了音频信号的存储和重放。例如，音乐和语音业已通过留声技术（例如唱盘播放机）、磁技术（例如卡式磁带）和数字技术（例如光盘）来记录和保存。随着音频存储技术的发展，需要克服许多难题来优化音频信号的质量和可存储性。

为了音乐信号的存档和宽带传输，无损重建成为比借助诸如 MP3 或 AAC 等在 MPEG 标准中定义的感性编码高效率压缩更为重要的特征。

虽然 DVD 音频和超级 CD 音频包括专利无损压缩方案，但是在内容持有者和广播公司当中需要一种开放和综合性的压缩方案。响应于这种需要，一种新的无损编码方案已经成为 MPEG-4 音频标准的延伸。无损音频编码法由于原始信号的完美重建而实现了没有任何质量损失的数字音频数据压缩。

发明内容

本发明涉及处理音频信号的方法。

在一个实施例中，音频信号的一个帧中的至少一个声道被细分成多个块以使这些块中的至少两个具有不同的长度，并且生成指示该声道被细分成若干块的信息。

根据一个实施例，根据细分分级结构来细分声道。所述细分分级结构具有一个以上的等级，并且每个等级关联于一个不同的块长度。例如，细分分级结构的上级关联于是与下级相关联的块长度的两倍的块长度。

在一个实施例中，如果声道具有长度 N ，则声道被细分成多个块以使每个块具有 $N/2$ 、 $N/4$ 、 $N/8$ 、 $N/16$ 、 $N/32$ 中的一个的长度。

在一个实施例中，生成信息以使该信息长度取决于细分分级结构中的级数。例如，可生成信息以使该信息包括数个信息比特，并且这些信息比特指示

声道被细分成若干块。更具体地说，每个信息比特关联于细分分级结构中的一级并可关联于相关联的一级上的一个块，并且每个信息比特指示相关联的块是否被细分。

在一个实施例中，该方法还包括基于之前的数据采样预测声道中的当前数据采样。用于预测的之前的数据采样的数目被称为预测阶数。当前数据采样的残差基于预测出的数据采样来获得。

在一个实施例中，该预测是通过当之前的数据采样可用时渐进地增大预测阶数至合需的预测阶数来执行的。例如，可对随机存取帧执行该渐进预测过程，所述随机存取帧是以无需用到之前的帧就可对其解码的方式编码的帧。

在另一实施例中，每个块的合需的预测阶数是基于最大允许预测阶数和块长度来确定的。

本发明进一步涉及编码音频信号的方法和装置，以及解码音频信号的方法和装置。

附图简要说明

包括于此以提供对本发明的进一步理解、并被结合在本申请中且构成其一部分的附图示出本发明的实施方式，其与说明书一起可用来解释本发明的原理。在附图中：

图 1 是根据本发明一个实施方式的编码器的示例图。

图 2 是根据本发明一个实施方式的解码器的示例图。

图 3 是根据本发明一个实施方式的压缩的 M-声道文件的比特流结构的示例图。

图 4 是根据本发明一个实施方式的分级块切换方法的概念图的示例图。

图 5 是块切换示例及相应的块切换信息代码的示例图。

图 6 是根据本发明实施方式的多个声道的块切换方法的示例图。

具体实施方式

下面将详细参考本发明的较佳实施方式，其具体示例图示于附图中。只要有可能，即在所有附图中使用相同的附图标记表示相同或相似的部件。

在对本发明进行叙述之前，应当指出的是本发明中揭示的大多数术语对应于本领域内公知的一般术语，但部分术语是由申请人根据需要选择的，并且将

在本发明下文的描述中予以揭示。因此，由申请人定义的术语优选基于它们在本发明中的含义来理解。

在无损音频编码方法中，由于编码过程必须是完全可逆而不会有信息损失的，因此编码器和解码器两者的若干部件必须以确定性的方式来实现。

编码解码器结构

图 1 是根据本发明一个的编码器 1 的示例图。

分割部件 100 将输入的音频数据分割成若干帧。在一帧内，每个声道还可进一步被细分成若干个音频采样块以做进一步处理。缓冲器 110 存储由分割部件 100 分割后的块和/或帧采样。

系数估算部件 120 针对每个块估算最佳的一组系数值。系数的数目，即预测器的阶数也可以自适应地做出选择。系数估算部件 120 针对数字音频数据块计算一组部分自相关系数（parcor）值。部分自相关系数值指示预测器系数的部分自相关系数表示。量化部件 130 将该组部分自相关系数值量化。

第一熵编码部件 140 通过从部分自相关系数值减去一个偏移值来计算出部分自相关系数残差值，并使用由熵参数所确定的熵代码对部分自相关系数的残差值进行编码，其中偏移值和熵参数选自最佳表。最优表是基于数字音频数据块的采样率从多个表中选择的。这多个表是分别对多个采样率范围预定义的以实现为了传输而进行的数字音频数据的最优压缩。

系数转换部件 150 将量化的部分自相关系数转换成线性预测编码（LPC）系数。预测器 160 使用线性预测编码系数从存储在缓冲器 110 中的之前的原始采样估算当前预测值。减法器 170 使用存储在缓冲器 110 中的数字音频数据的原始值和预测器 160 中估算出的预测值计算数字音频数据块的预测残差。

第二熵编码部件 180 使用不同的熵代码编码预测残差并生成代码索引。所选代码的索引将作为辅助信息发送。第二熵编码部件 180 可使用具有不同复杂度的两种可供选择的编码技术之一对预测残差进行编码。一种编码技术是公知的 Golomb-Rice 编码（在下文中简称为 Rice 代码）法而另一种是公知的分块 Gilbert-Moore 代码（在下文中简称为 BGMC）法。” Rice 代码复杂度低但仍然是高效率的。BGMC 算术编码方案以比 Rice 代码复杂度稍高为代价提供更好的压缩。

最后，多路复用部件 190 将编码的预测残差、代码索引、编码的部分自相

关系数残差值和其它附加信息进行多路复用形成压缩的比特流。编码器 1 还提供循环冗余校验 (CRC) 校验和, 它主要是供解码器验证解码的数据。在编码器方面, CRC 可用来确保压缩的数据是能够进行无损解码的。

其它编码选项包括灵活块切换方案、随机存取和联合声道编码。编码器 1 可使用这些选项提供若干具有不同复杂度的压缩等级。将联合声道编码用于利用立体声或多声道信号的各声道之间的相关性这可通过在能够比原始信道之一更高效地编码两个声道之间的差异的片段 (segments) 中编码这种差异来实现。这些编码选项将在对根据本发明的示例性解码器进行说明之后更为详细地予以说明。

图 2 是根据本发明的解码器 2 的示例图。更特别地, 图 2 示出由于不必执行任何适应性调整因而复杂度显著低于编码器的无损音频信号解码器。

多路分解部件 200 接收音频信号并将经编码的数字音频数据块的预测残差、代码索引、经编码的部分相关残差值和其它附加信息多路分解。第一熵解码部件 210 使用由熵参数定义的熵代码对部分自相关系数残差值进行解码并通过将偏移值加至解码的部分自相关系数残差值来计算出一组部分自相关系数值; 其中的偏移值和熵参数选自一个表, 该表是由解码器基于数字音频数据块的采样率从多个表中选择的。第二熵解码部件 220 使用代码索引对经多路分解编码的预测残差进行解码。系数转换部件 230 将熵解码的部分相关值转换成 LPC 系数。预测器 240 使用 LPC 系数估算数字音频数据块的预测残差。加法器 250 将经解码的预测残差加至估算的预测残差以获得数字音频数据的原始块。组装部件 260 将经解码的块数据组装成帧数据。

因此, 解码器 2 将经编码的预测残差和部分相关残差值解码, 将部分相关残差值转换成 LPC 系数, 并应用反向预测滤波器来计算无损重建信号。解码器 2 的计算工作量取决于由编码器 1 选择的预测阶数。在大多数情形下, 实时解码即使在低端系统中也是可能的。

图 3 是根据本发明的包括多个声道 (例如 M 个声道) 的压缩音频信号的比特流结构的示例图。

比特流由至少一个包括多个声道 (例如 M 个声道) 的音频帧构成。比特流配置句法 (见下面的表 6) 中的 “channels” 字段指示声道数。每个声道利用根据本发明的块切换方案细分成多个块, 这将在后面详细说明。每个细分的块具有不同的大小并包括根据图 1 的编码法的编码数据。例如, 一个细分块内的

编码数据包含代码索引、预测阶数 K 、预测器系数和编码的残差值。如果在声道对之间使用联合编码法，则这两个声道的块分割是等同的，而且这些块以交织方式存储。比特流配置句法（表 6）中的“js_stereo”字段指示联合立体声（声道差）是开启的还是关闭的，而 frame_data 句法（见下面的表 7）中的“js_switch”字段指示是否选择联合立体声（声道差）。否则，每个声道的块分割是独立的。

下面将参照附图及其后的句法详细地说明前面提到的块切换、随机存取、预测和熵编码选项。

块切换

本发明的一个方面涉及在使用实际编码方案之前将每个声道细分成多个块。在下文中，根据本发明的块分割（或细分）方法将被称为“块切换方法”。

分级块切换

图 4 是根据本发明的分级块切换方法的概念图的示例图。例如，图 4 示出将一个声道按分级方式细分成 32 个块的方法。当在单个帧中提供多个声道时，每个声道被细分（或分割）成最多达 32 个块，并且每个声道的细分块配置成一个帧。因此，根据本发明的块切换方法由图 1 所示的分割部件 100 执行。此外，如上所述，预测和熵编码在细分的块单元上执行。

一般而言，传统的音频无损编码（ALS）包括相对简单的块切换机制。每一个 N 个采样的声道或者使用一个全长块（ $N_b=N$ ）进行编码，或者使用四个长度 $N_b=N/4$ 的块（例如 1:4 切换）进行编码，其中同一块分割法适用于所有声道。在某些情形下，该方案会具有某些局限性。例如，尽管只有 1:1 或 1:4 切换是能用的，但其他切换法（例如 1:2、1:8 及其组合）在某些情形下却效率更高。另外在传统 ALS 中，对所有声道以等同的方式执行切换，虽然不同声道或许会从不同的切换法受益（如果声道并不相关时则尤为如此）。

因此，根据本发明实施方式的块切换方法提供相对灵活的块切换方案，其中一个帧的每个声道可按分级方式被细分成多个块。例如，图 4 示出能以分级方式细分成最多达 32 个块的声道。在根据所给出的实施方式的声道内， $N_b=N$ 、 $N/2$ 、 $N/4$ 、 $N/8$ 、 $N/16$ 和 $N/32$ 的块的任意组合是可能的，只要每个块是通过双倍长度的上级块进行细分产生的即可。例如，如图 4 中的例子所示，分割成

$N/4 + N/4 + N/2$ 是可能的, 而分割成 $N/4 + N/2 + N/4$ 是不行的 (例如在下面描述的在图 5(e) 和图 5 中示出的块切换示例)。换言之, 声道被分成这多个块从而使每个块的长度等于 $N/(m^i)$ 中的一个值, 其中 $i=1, 2, \dots, p$, N 是声道的长度, m 是大于或等于 2 的一个整数, 而 p 表示在细分分级结构中的级数。

因此, 在本发明的实施方式中, 比特流包括指示块切换等级的信息以及指示块切换结果的信息。这里, 与块切换相关的信息被包含在解码过程中使用的句法中, 这将在下面进行描述。

例如, 作出设定以使块切换处理后生成的最小块大小为 $N_b = N/32$ 。然而, 这种设定仅仅是为简化本发明说明的一个示例。因此, 根据本发明的设定不限于这一种设定。

更具体地, 当最小块大小为 $N_b = N/32$ 时, 这表示块切换处理已按分级方式进行了 5 次, 故将其称为 5 级块切换。或者, 当最小块大小为 $N_b = N/16$ 时, 这表示块切换处理已按分级方式进行了 4 次, 故将其称为 4 级块切换。类似地, 当最小块大小为 $N_b = N/8$ 时, 这表示块切换处理已按分级方式进行了 3 次, 故将其称为 3 级块切换。而当最小块大小为 $N_b = N/4$ 时, 这表示块切换处理已按分级方式进行了 2 次, 故将其称为 2 级块切换。当最小块大小为 $N_b = N/2$ 时, 这表示块切换处理已按分级方式进行了 1 次, 故将其称为 1 级块切换。最后, 当最小块大小为 $N_b = N$ 时, 这表示未执行分级块切换处理, 故将其称为 0 级块切换。

在本发明的实施方式中, 指示块切换等级的信息被称为第一块切换信息。例如, 第一块切换信息可由表 6 中所示的句法中一个 2 比特字段 “block_switching” 表示, 这将在后面的过程中予以说明。更具体地, “block_switching=00” 表示 0 级, “block_switching=01” 表示 1 级至 3 级中的任何一个, “block_switching=10” 表示 4 级, 而 “block_switching=11” 表示 5 级。

另外, 指示根据上述块切换等级对每个等级所执行的块切换结果的信息在这些实施方式中被称为第二块切换信息。这里, 第二块切换信息可由 “bs_info” 字段表示, 该字段在表 7 所示的语法中以 8 比特、16 比特和 32 比特中的任何一个表示。更具体地说, 如果 “block_switching=01” (表示 1 级至 3 级的任何一个), 则 “bs_info” 由 8 比特表示。如果 block_switching=10 (表示 04 级), 则 bs_info 由 016 比特表示。换言之, 最多达 4 级的块切换结果可用

16 比特指示。此外，如果“block_switching=11”（表示 5 级），则“bs_info”用 32 比特表示。换言之，最长达 5 级的块切换结果可用 32 比特指示。最后，如果 block_switching=00（表示未进行块切换），则不发送 bs_info。这表示一个声道配置成一个块。

分配给第二块切换信息的总比特数是基于第一块切换信息的等级值确定的。这可能会导致最终比特率降低。在下面的表 1 中简要描述了第一块切换信息和第二块切换信息之间的关系。

表 1：块切换等级

最大等级数	最小 N_B	“bs_info”的字节数
0 ("block_switching=00")	N	0
1 ("block_switching=01")	$N/2$	1 (=8 比特)
2 ("block_switching=01")	$N/4$	1 (=8 比特)
3 ("block_switching=01")	$N/8$	1 (=8 比特)
4 ("block_switching=10")	$N/16$	2 (=16 比特)
5 ("block_switching=11")	$N/32$	4 (=32 比特)

下面将详细描述配置（或映射）第二块切换信息（bs_info）内每个比特的方法的一个实施方式。

bs_info 字段根据上述实施方式可包括最长达 4 个字节。关于 1 级至 5 级的比特映射可以是 [(0)1223333 44444444 55555555 55555555]。可保留第一比特以指示是独立块切换还是同步块切换，这将在后面的独立/同步块切换一节中更为详细地描述。图 5(a)–5(f)示出可发生 3 级块切换的一个声道的不同块切换示例。因此，在这些示例中，最小块长度为 $N_B=N/8$ ，且 bs_info 由一个字节构成。从最大块长度 $N_B=N$ 开始，如果块被进一步细分，则 bs_info 的比特被置位。例如，在图 5(a)中，根本不存在细分，因此“bs_info”为 (0)000 0000。在图 5(b)中，帧被细分 ((0)1……) 而长度为 $N/2$ 的第二块被进一步分 ((0)101……) 成两个长度 $N/4$ 的块；因此 bs_info 为 (0)1010 0000。

在图 5(c)中，帧被细分 ((0)1...)，且只有长度为 $N/2$ 的第一块被进一步分 ((0)110...) 成为两个长度为 $N/4$ 的块；因此“bs_info”为 (0)1100 0000。在图 5(d)中，帧被细分 ((0)1……)，长度为 $N/2$ 的第一块和第二块被进一步分

((0)111……)成长度为 $N/4$ 的两个块，并且只有长度为 $N/4$ 的第二块被进一步分((0)11101……)成长度为 $N/8$ 的两个块；因此“bs_info”为(0)111 0100。

如上所述，图 5(e)和 5(f)中的示例表示不被允许的块切换的情形，这是因为图 5(e)中的 $N/2$ 块和图 5(f)中的第一个 $N/4$ 块不可能通过细分前一级的块来获得的。

独立/同步块切换

图 6(a)–6(c)是根据本发明实施方式的块切换的示例图。

更具体地，图 6(a)示出未对声道 1、2 和 3 执行块切换的示例。图 6(b)示出的是其中两个声道（声道 1 和 2）配置成一个声道对、且在声道 1 和声道 2 中同步地执行块切换的示例。在本例中还应用了交织处理。图 6(c)示出的是其中两个声道（声道 1 和 2）配置成一个声道对、且独立地对声道 1 和声道 2 执行块切换的示例。在本发明中，“声道对”指两个任意的音频声道。关于哪些声道组成声道对的决定可由编码器自动做出或由用户人工做出。

在独立块切换中，尽管在所有声道中每个声道的长度可以是相同的，但可对每个声道个别地执行块切换。即，如图 6(c)所示，各声道可以不同方式分成块。如果一个声道对的两个声道彼此相关并且使用差分编码，则该声道对的两个声道可被同步地进行块切换。在同步块切换中，各声道以相同方式进行块切换（即分成块）。图 6(b)示出这样的示例，并进一步示出这些块是可以被交织的。如果声道对的两个声道彼此不相关，则差分编码并无益处，因而不需要对声道同步地进行块切换。相反，独立地切换声道可能更合适。

此外，根据本发明的另一实施方式，所描述的独立或同步块切换方法可应用于声道数大于或等于 3 的多声道组。例如，如果该多声道组的所有声道彼此相关，则可以同步切换多声道组的所有声道。另一方面，如果多声道组的所有声道彼此不相关，则可以独立地切换多声道组的每个声道。

此外，“bs_info”字段被用作指示块切换结果的信息。另外，键 s_info 字段还被用作指示对配置成声道对的每个声道是独立地执行了块切换还是同步地执行了块切换的信息。”在这种情况下，如上所述，可使用“bs_info”字段中的特定比特（例如第一比特）。例如，如果声道对的两个声道彼此独立，则 bs_info 字段的第一比特被置为 1。另一方面，如果声道对的两个声道彼此同步，则 bs_info 字段的第一比特被置为 0。

下面将详细描述图 6(a)、6(b)和 6(c)。

参照图 6(a)，由于没有一个声道执行了块切换，因此不生成有关的“bs_info”。

参照图 6(b)，声道 1 和 2 构成一个声道对，其中两个声道彼此同步且同步执行了块切换。例如，在图 6(b)中，声道 1 和声道 2 两者都被分割成长度为 $N/4$ 的块，两者都具有相同的 bs_info “bs_info=(0)101 0000”。因此，可针对每个声道对发送一个 bs_info，这导致比特率下降。此外，如果声道对是同步的，则声道对中的每个块会被要求彼此交织。这种交织是有益的（或有利的）。例如，一个声道对内的一个声道的块（例如图 6(b)中的块 1.2）对两个声道的之前的块（例如图 6(b)中的块 1.1 和 2.1）都有依赖关系，因此这些之前的块应当在当前块之前就已可用。

参照图 6(c)，声道 1 和 2 配置成一个声道对。然而，在本例中，块切换是独立执行的。更具体地，声道 1 被分割成大小（或长度）达 $N/4$ 的块，并且 bs_info 是 “bs_info=(1)101 0000”。声道 2 被分割成大小达 $N/2$ 的块，并且 bs_info 是 “bs_info=(1)100 0000”。在图 6(c)所示例子中，在每个声道间独立地进行块切换，因此并不执行块之间的交织处理。换言之，对于独立地进行了块切换的声道，声道数据可单独编排。

联合声道编码

联合声道编码——也被称为联合立体声——可利用立体声信号的两个声道之间或多声道信号的任何两个声道之间的相关性。尽管独立地处理两个声道 $x_1(n)$ 和 $x_2(n)$ 更为直接，但利用声道之间相关性的简单方法是对差分信号进行编码：

$$d(n) = x_2(n) - x_1(n)$$

而不是对 $x_1(n)$ 或 $x_2(n)$ 进行编码。可通过对个体信号加以比较，根据哪两个信号能被最高效率地编码，在每个块中的 $x_1(n)$ 、 $x_2(n)$ 和 $d(n)$ 之间进行切换。这种用切换的差分编码实现的预测在两个声道彼此非常相似的情形中是有利的。在多声道素材的情形中，可由编码器重新编排声道以指派合适的声道对。

除了简单的差分编码，无损音频编解码器还支持更为复杂的利用多声道信号的任意声道之间的声道间冗余的方案。

随机存取

本发明涉及音频无损编码并能够支持随机存取。随机存取意味着对编码的音频信号任意部分的快速存取而无需浪费地对之前的各部分进行解码。这对采用压缩数据的查找、编辑或流送的应用是一个重要特征。为了实现随机存取，在随机存取单元内，编码器需要插入一个能在无需解码之前各帧的情况下进行解码的帧。插入的帧被称为“随机存取帧”。在此类随机存取帧中，没有任何来自之前各帧的采样可供用于预测。

下面将详细描述根据本发明的用于实现随机存取的信息。参照配置句法（表 6 所示），与随机存取有关的信息作为配置信息发送。例如，“random_access”字段被用作指示是否允许随机存取的信息，它可用 8 比特表示。此外，如果允许随机存取，则该 8 比特“random_access”字段指定配置成一个随机存取单元的帧数。例如，当“random_access = 0000 0000”时，不支持随机存取。换言之，当“random_access > 0”时，则支持随机存取。更具体地，当“random_access = 0000 0001”时，这指示配置成随机存取单元的帧数为 1。这表示在所有的帧单元中均允许随机存取。此外，当“random_access = 1111 1111”，这指示配置成随机存取单元的帧数为 255。因此，“random_access”信息对应于当前随机存取单元内的随机存取帧与下一随机存取单元中的随机存取帧之间的距离。在本发明中，所述距离用帧数表达。

一个 32 比特的“ra_unit_size”字段被包含在比特流中并且被发送。”在本发明中，“ra_unit_size”字段指示以字节计从当前随机存取帧到下一随机存取帧的大小。因此，“ra_unit_size”字段或者包含在配置句法（表 6）中或者包含在帧数据句法（表 7）中。配置句法（表 6）还可包括指示“ra_unit_size”信息在比特流中的存储位置的信息。该信息被表示为 2 比特的“ra_flag”字段。更具体地，例如，当“ra_flag = 00”时，这表示“ra_unit_size”信息未存储在比特流中。当“ra_flag = 01”时，这表示“ra_unit_size”信息被存储在比特流内的帧数据句法（表 7）中。

此外，当“ra_flag = 10”时，“ra_unit_size”信息被存储在比特流的配置句法（表 6）。如果“ra_unit_size”信息被包含在配置句法中，则这表示“ra_unit_size”信息只在比特流上发送一次并且被同等地应用于所有随机存取单元。或者，如果“ra_unit_size”信息包含帧数据句法中，则这表示当前随机存取单元内的随机存取帧和下一随机存取单元内的随机存取帧内之间的

距离。因此，针对比特流中的每一个随机存取单元发送“ra_unit_size”信息。

因此，配置句法（表 6）内的“random_access”字段也可被称为第一通用消息。另外，“ra_flag”字段也可被称为第二通用消息。在本发明的这个方面中，音频信号包括配置信息和多个随机存取单元，每个随机存取单元含有一个或多个音频数据帧，所述音频数据帧中的一个为随机存取帧，其中所述配置信息包括指示诸帧中的两相邻随机存取帧之间的距离的第一通用信息、以及指示每个随机存取单元的随机存取单元大小信息被存储在哪里的第二通用信息。随机存取单元大小信息指示以字节计两相邻随机存取帧之间的距离。

或者，在本发明的这个方面，一种解码音频信号的方法包括：接收具有配置信息和多个随机存取单元的音频信号，每个随机存取单元含有一个或多个音频数据帧，所述音频数据帧中的一个为随机存取帧；从配置信息读取第一通用信息，所述第一通用信息指示诸帧中两相邻随机存取帧之间的距离；以及从配置信息读取第二通用信息，所述第二通用信息指示每个随机存取单元的随机存取大小信息被存储在哪里，而随机存取单元大小信息指示以字节计两相邻随机存取帧之间的距离。

声道配置

如图 3 所示，音频信号包括根据本发明的多声道信息。例如，每个声道可按与音频扬声器的位置一一对应的关系来映射。配置句法（下面的表 6）包括声道配置信息，它被表示为 16 比特的“chan_config_info”字段和 16 比特的“channels”字段。“chan_config_info”字段包括将声道映射到扬声器位置的信息，而 16 比特的“channels”字段包括指示声道总数的信息。例如，当“channels”字段等于“0”时，这表示声道对应于单声道。当“channels”字段等于“1”时，这表示这个声道对应于立体声声道中的一个。另外，当“channels”字段等于或大于“2”时，这表示这个声道对应于多声道中的一个。

下面的表 2 示出配置成“chan_config_info”字段的每个比特以及与之对应的各个声道的示例。更具体地，当所发送的比特流中存在相应声道时，“chan_config_info”字段内的相应比特被置为“1”。或者，当所发送的比特流中不存在相应声道时，“chan_config_info”字段内的相应比特被置为“0”。本发明还包括指示配置句法（表 6）内是否存在“chan_config_info”

的信息。该信息被表示为 1 比特的“chan_config”标志。更具体地，“chan_config = 0”指示“chan_config_info”字段不存在。而“chan_config = 1”指示“chan_config_info”字段存在。因此，当“chan_config = 0”时，这表示“chan_config_info”字段不是在配置句法（表 6）内新定义的。

表 2：声道配置

扬声器位置	缩写	chan_config_info 中比特位置
左	L	1
右	R	2
左后	Lr	3
右后	Rr	4
左侧	Ls	5
右侧	Rs	6
中置	C	7
后中置/环绕	S	8
低频效果	LPE	9
左混频	LO	10
右混频	RO	11
单声道混频	M	12
（保留）		13-16

帧长度

如图 3 所示，根据本发明的音频信号包括多个声道或多声道。因此，当执行编码时，关于配置成一帧的多声道的数目的信息以及关于每个声道的采样数的信息被插入到比特流中并被发送。参照配置句法（表 6），32 比特的“samples”字段被用作指示配置成每个声道的音频数据采样总数的信息。此外，16 比特的“frame_length”（帧长度）字段被用作指示相应帧内每个声道的采样数的信息。

此外，“frame_length”字段的 16 比特值是由编码器所使用的值确定的，并且被称为用户定义值。换言之，用户定义值不是固定值，而是在编码过程中任意确定的值。

因此，在解码过程中，当通过图 2 所示的多路分解部件 200 接收到比特流时，应当首先获取每个声道的帧数。该值是根据下面所示的算法得到的。

```
frames = samples / frame_length;
rest = samples % frame_length; if (rest)
{
    frames++;
    frlen_last = rest;
}
else
    frlen_last = frame_length;
```

更具体地，每个声道的帧总数是通过将经由比特流发送的“samples”字段确定的每个声道的采样总数除以由“frame_length”字段确定的每个声道的一个帧内的采样数来计算得到的。例如，当由“samples”字段确定的采样总数恰好是由“frame_length”字段确定的每个帧内的采样数的倍数时，则该倍数成为帧总数。然而，如果由“samples”字段确定的采样总数并非恰好是由“frame_length”字段确定的采样数的倍数，而是存在余数（或残数），则总帧数比倍数增加“1”。此外，最末帧的采样数（frlen_last）被确定为该余数（或残数）。这表示仅最末帧的采样数与其之前的帧不同。通过如上所述地在编码器和解码器之间定义一套标准化的规则，编码器就可自由地确定并发送每个声道的采样总数（“samples”字段）以及每个声道的一个帧内的采样数（“frame_length”字段）。此外，解码器可通过对所发送信息上使用上述算法而精确地确定要用于解码的每个声道的帧数。

线性预测

在本发明中，应用线性预测以实现无损音频编码。图 1 所示的预测器 160 包括至少一个或多个滤波器系数以从之前的采样值预测当前的采样值。随后，第二熵编码部件 180 对与预测值和原始值之差相对应的残差值执行熵编码。另外，应用于预测器 160 的每个块的预测器系数值是作为最优值从系数估算部件 120 选择的。此外，预测器系数值由第一熵编码部件 140 进行熵编码处理。由第一熵编码部件 140 和第二熵编码部件 180 编码的数据作为比特流的一部分由多路复用部件 190 插入且随后被发送。

下面将详细描述根据本发明的执行线性预测的方法。

用 FIR 滤波器进行的预测

线性预测在许多应用场合中被用于实现语音和音频信号处理。在下文中，基于有限冲激响应（FIR）滤波器描述预测器 160 的示例性操作。然而，本例明显不是对本发明范围的限制。时间离散信号 $x(n)$ 的当前采样可根据之前的采样 $x(n-k)$ 大致地预测出。预测由以下方程式给出。

$$\hat{x}(n) = \sum_{k=1}^K h_k * x(n-k),$$

其中 K 是预测器的阶数。如果预测的采样接近于原始采样，则如下所示的残差：

$$e(n) = x(n) - \hat{x}(n)$$

的方差比 $x(n)$ 本身的更小，因此能更高效地编码 $e(n)$ 。

从输入采样的片段估算预测器系数然后再对该片段进行滤波处理的程序被称为前向自适应。在这种情况下，应当发送这些系数。另一方面，如果是从之前已处理的片段或采样（例如从残差）估算系数，则称为后向自适应。后向自适应程序的优点在于不需要发送系数，因为估算系数所需的数据对于解码器也是可用的。

10 阶左右的前向自适应预测方法被广泛地用于语音编码，并且可同样适用于无损音频编码。大多数前向自适应无损预测方案的最大阶数仍然相当小，例如 $K=32$ 。一个例外是超级音频 CD 专用的 1 比特无损编解码器，它使用高达 128 的预测阶数。

另一方面，具有几百个系数的后向自适应 FIR 滤波器通用于许多领域，例如声道均衡和回波抵消。这些系统大多数是基于 LMS 算法或其变型的，这些算法也被推荐用于无损音频编码。并非必须要将预测器系数作为辅助信息发送，因此它们的数目对数据速率不产生影响。然而，后向自适应的编解码器的缺点在于：必须在编码器和解码器两者中作出自适应，这使解码器明显比前向自适应情况下的解码器更为复杂。

前向自适应预测

作为本发明的示例性实施方式，前向自适应预测将作为一个示例在本文的

描述中给出。在前向自适应线性预测中，一般使用自相关方法或协方差方法由系数估算部件 120 估算每个块的最优预测器系数 h_k （在残差方差最小化的意义上）。使用传统的 Levinson-Durbin 算法的自相关方法的额外优点是提供了一种迭代式自适应调整预测器阶数的简单方法。此外，该算法本身也计算相应的部分自相关系数。

前向自适应预测的另一方面是确定合适的预测阶数。阶数增大使预测误差的方差减小，这导致残差的比特率 R_c 变小。另一方面，预测器系数的比特率 R_e 随着要被发送的系数的数目而提高。因此，任务是找到使总比特率最小化的最优阶数。这可通过关于预测阶数 K 使下面的等式最小化来表达：

$$R_{\text{total}} \{K\} = R_e \{K\} + R_c(K),$$

K 是预测阶数。由于预测增益随阶数升高而单调上升，因此 R_e 随着 K 值而下降。另一方面，由于要发送的系数的数目增加，因此 R_c 随 K 值单调上升。

搜索最优阶数可由系数估算部件 120 高效率地执行，所述系数估算部件 120 用递归方式确定阶数递增的所有预测器。对于每个阶数，计算完整的一组预测器系数。另外，可推导出相应残差的方差 σ_e^2 ，从而得到残差的预期比特率的估算值。在每次迭代过程中——即针对每个预测阶数——在确定各系数的比特率的同时还可确定总比特率。最优阶数在总比特率不再减小的点找到。

尽管从上述方程式可以清楚知道系数比特率对总比特率有直接的影响，但是， R_c 缓慢的增长也使得 R_{total} 的最小值移至较高的阶数（其中 R_e 同样较小），这可产生更好的压缩。因此，预测器系数的高效率但仍准确的量化在实现最大压缩中发挥着重要作用。

预测阶数

在本发明中，确定了预测阶数 K ，预测阶数 K 决定用于进行线性预测的预测器系数的数目。预测阶数 K 也是由系数估算部件 120 予以确定。在本发明中，关于所确定的预测阶数的信息被包含在比特流中并随后被发送。

配置句法（表 6）包括与预测阶数 K 有关的信息。例如，1 比特至 10 比特的“max_order”字段对应于指示最大阶数值的信息。1 比特至 10 比特的“max_order”字段的最大值是 $K=1023$ （例如 10 比特）。作为与预测阶数 K 有关的另一信息，配置句法（表 6）包括 1 比特的“adapt_order”字段，它指示每个块是否存在最优阶数。例如，当“adapt_order=1”时，应当给每个块

提供最优阶数。在 block_data 句法（表 8）中，最优阶数作为 1 比特至 10 比特的“opt_order”字段提供。此外，当“adapt_order=0”时，则不对每个块提供单独的最优阶数。在这种情况下，“max_order”字段即成为应用于所有块的最终阶数。

最优阶数（opt_order）是基于 max_order 字段值和相应块的大小（ N_b ）确定的。更具体地，例如当 max_order 被确定为 $K_{max}=10$ 并且“adapt_order=1”时，则可考虑相应块的大小确定每个块的 opt_order。在某些情况下，大于 max_order（ $K_{max}=10$ ）的 opt_order 值是可能的。

特别地，本发明涉及较高的预测阶数。根据本发明的实施方式，在没有分级块切换的情形中，在长和短的块长度之间可能是 4 倍因数的关系（例如 4096 与 1024 或 8192 与 2048）。另一方面，在采用了分级块切换的实施方式中，这个因数可以提高（例如高达 32），以使范围更大（例如从 16384 下至 512 或甚至在高采样率下从 32768 至 1024）。

在执行了分级块切换的实施方式中，为了更好地使用非常长的块，可采用更高的最大预测阶数。最大阶数可以是 $K_{max}=1023$ 。在这些实施方式中， K_{max} 可由块长度 N_b 界定，例如 $K_{max} < N_b/8$ （例如当 $N_b=2048$ 时， $K_{max}=255$ ）。因此，使用 $K_{max}=1023$ 需要至少 $N_b=8192$ 的块长度。在这些实施方式中，配置句法（表 6）中的“max_order”字段可高达 10 比特而 block_data 句法（表 8）中的“opt_order”字段同样可高达 10 比特。具体块中的实际比特数可取决于一个块所允许的最大阶数。如果块是短块，则本地预测阶数可小于全局预测阶数。在本发明中，本地预测阶数是通过考虑相应块长度 N_b 确定的，而全局预测阶数是通过配置句法中的“max_order” K_{max} 确定的。例如，如果 $K_{max}=1023$ ，但 $N_b=2048$ ，则由于本地预测阶数为 255，因此“opt_order”字段被确定为 8 比特（而不是 10 比特）。

更具体地，可基于下面的方程式确定 opt_order：

$opt_order = \min(\text{全局预测阶数}, \text{本地预测阶数})$ ；

另外，全局和本地预测阶数可通过下面的方程式确定：

全局预测阶数 = $\text{ceil}(\log_2(\text{最大预测阶数} + 1))$

本地预测阶数 = $\max(\text{ceil}(\log_2((N_b \gg 3) - 1)), 1)$

在这些实施方式中，预测了来自一个声道的细分块的数据采样。使用之前块的最末 K 个采样预测当前块的第一采样。K 值是基于从上述方程式推导出的

opt_order 确定的。

如果当前块是声道的第一个块，则不使用来自之前块的采样。在这种情形中，采用的是渐进阶数预测。例如，假设相应块的 opt_order 值为 $K=5$ ，则该块中的第一采样不执行预测。该块的第二采样使用该块的第一采样执行预测（如同 $K=1$ ），该块的第三采样使用该块的第一采样和第二采样执行预测（如同 $K=2$ ）等。因此，从第六采样开始以及对于这之后的采样，根据 $K=5$ 的 opt_order 执行预测。如上所述，预测阶数从 $K=1$ 渐进地增加至 $K=5$ 。

当在随机存取帧中使用时，上述渐进阶数型预测是非常有利的。由于随机存取帧对应于随机存取单元的基准帧，因此随机存取帧不是通过使用之前的帧采样执行预测。即，这种渐进预测技术在随机存取帧的开头处就可应用。

预测器系数的量化

上述预测器系数在图 1 的量化部件 130 中量化。由于即便很小的量化误差也会导致大大偏离最优预测滤波器所需的频谱特性，因此预测系数 h_k 的直接量化对发送而言不是非常高效率的。为此，预测器系数的量化是基于可由系数估算部件 120 计算得到的部分自相关（反射）系数 r_k 。例如，如上所述，系数估算部件 120 是使用传统 Levinson-Durbin 算法处理的。

头两个部分自相关系数（相应地为 γ_1 和 γ_2 ）通过使用下面的函数被量化：

$$a_1 = \left\lfloor 64 \left(-1 + \sqrt{2\sqrt{\gamma_1 + 1}} \right) \right\rfloor;$$

$$a_2 = \left\lfloor 64 \left(-1 + \sqrt{2\sqrt{-\gamma_2 + 1}} \right) \right\rfloor;$$

而其余系数是使用简单的 7 比特的均匀量化器量化的：

$$a_k = \lfloor 64 \gamma_k \rfloor; \quad (k > 2).$$

在所有情况下，所得量化值 a_k 被约束在范围 $[-64, 63]$ 内。

熵编码

如图 1 所示，在本发明中应用了两种类型的熵编码。更具体地，第一熵编码部件 140 被用于编码上述预测器系数。另外，第二熵编码部件 180 被用来编码上述音频原始采样和音频残差采样。在下文中将详细说明这两种类型的熵编

码。

预测器系数的第一熵编码

相关技术的 Rice 代码被用作根据本发明的第一熵编码方法。例如，量化系数 a_k 的发送是通过生成残差值执行的：

$$\delta_k = a_k - \text{offset}_k,$$

这些残差值进而是通过使用第一熵编码部件 140——例如用 Rice 代码方法来编码的。该过程中使用的 Rice 代码的相应偏移和参数可从以下表 3、4 和 5 所示的诸组中的一个以全局方式进行选择。表索引（即 2 比特的“coef_table”）在配置句法（表 6）中指出。如果“coef_table=11”，则表示未应用熵编码，并且量化的系数各自以 7 比特发送。在这种情形中，偏移始终是 -64 以获得被约束于 [0, 127] 的无符号值 $\delta_k = a_k + 64$ 。相反，如果“coeff_table=00”，则选择下面的表 3，而如果“coeff_table=01”，则选择下面的表 4。最后，如果“coeff_table=11”，则选择表 5。当在图 2 的解码器中接收到这些经量化的系数时，第一熵解码部件 220 通过使用将残差值 δ_k 与偏移结合以生成部分自相关系数 a_k 的量化索引的过程来重建预测器系数：

$$a_k = \delta_k + \text{offset}_k.$$

其后，通过使用下面的方程式执行头两个系数（ γ_1 和 γ_2 ）的重建：

$$\begin{aligned} \text{par}_1 &= \lfloor \hat{\gamma}_1 2^Q \rfloor = \Gamma(a_1); \\ \text{par}_2 &= \lfloor \hat{\gamma}_2 2^Q \rfloor = -\Gamma(a_2); \end{aligned}$$

其中 2^Q 表示重构的系数的整数表示所需的恒量（ $Q=20$ ）比例因数，而 $\Gamma(\cdot)$ 是根据经验确定的映射表（未示出，因为映射表会根据实现而变化）。

因此，是根据采样频率提供用于进行第一熵编码的这三种类型的系数表。例如，采样频率可被分成 48kHz、96kHz 和 192kHz。这里，三个表 3、4、5 中的每一个分别提供给每种采样频率。

可对整个文件选择三个不同的表中的一个，而不是使用单个表。一般应当根据采样率来选择表。对于 44.1kHz 的素材，本发明的申请人推荐使用 48kHz 表。然而，一般而言，也可按其它准则来选择表。

表 3：编码量化系数（48kHz）使用的 Rice 代码参数。

系数#	偏移	Rice 参数
-----	----	---------

1	-52	4
2	-29	5
3	-31	4
4	19	4
5	-16	4
6	12	3
7	-7	3
8	9	3
9	-5	3
10	6	3
11	-4	3
12	3	3
13	-3	2
14	3	2
15	-2	2
16	3	2
17	-1	2
18	2	2
19	-1	2
20	2	2
$2k-1, k > 10$	0	2
$2k, k > 10$	1	2

表 4: 编码量化系数 (96kHz) 使用的 Rice 代码参数

系数#	偏移	Rice 参数
1	-58	3
2	-42	4
3	-46	4
4	37	5
5	-36	4

6	29	4
7	-29	4
8	25	4
9	-23	4
10	20	4
11	-17	4
12	16	4
13	-12	4
14	12	3
15	-10	4
16	7	3
17	-4	4
18	3	3
19	-1	3
20	1	3
$2k-1, k > 10$	0	2
$2k, k > 10$	1	2

表 5: 编码量化系数 (192kHz) 使用的 Rice 代码参数

系数#	偏移	Rice 参数
1	-59	3
2	-45	5
3	-50	4
4	38	4
5	-39	4
6	32	4
7	-30	4
8	25	3
9	-23	3
10	20	3

11	-20	3
12	16	3
13	-13	3
14	10	3
15	-7	3
16	3	3
17	0	3
18	-1	3
19	2	3
20	-1	2
$2k-1, k > 10$	0	2
$2k, k > 10$	1	2

残差的第二熵编码

本发明包含应用于图 1 的第二熵编码部件 180 的编码方法的两种不同的模式，这将在下面予以详细说明。

在简单模式中，使用 Rice 代码对残差值 $e(n)$ 进行熵编码。对于每个块，或者可使用同一 Rice 代码编码所有的值，或者可将块进一步分成四个部分，每个部分用一不同的 Rice 代码编码。如图 1 所示，发送所应用的代码的索引。由于存在不同的方法确定给定的一组数据的最优 Rice 代码，因此由编码器根据残差的统计结果选择合适的代码。

或者，编码器可利用 BGMC 模式使用更为复杂和高效率的编码方案。在 BGMC 模式中，残差的编码是通过将分布划分成两个类别实现的。这两种类型包括属于分布的中心区域 $|e(n)| < e_{\max}$ 的残差，以及属于其尾部的残差。尾部的余数仅仅是被重新居中 (re-centered) (即对于 $e(n) > e_{\max}$ ，提供 $e_t(n) = e(n) - e_{\max}$) 并使用如上所述的 Rice 代码编码。然而，为了编码处于分布中心的残差，BGMC 首先将残差分成 LSB 和 MSB 分量，随后 BGMC 使用块 Gilbert_Moore (算术) 代码编码 MSB。最后，BGMC 使用直接固定长度代码发送 LSB。可以对参数 e_{\max} 和直接发送的 LSB 的数目加以选择，使它们仅些微地影响这种方案的编码效率，同时使编码的复杂度明显降低。

根据本发明的配置句法（表 6）和 `block_data` 句法（表 8）包括与 Rice 代码和 BGMC 代码的编码有关的信息。现在对这种信息进行详细说明。

配置句法（表 6）首先包括一个 1 比特的“`bgmc_mode`”字段。例如，“`bgmc_mode=0`”表示 Rice 代码，“`bgmc_mode=1`”表示 BGMC 代码。配置句法（表 6）还包括一个 1 比特的“`sb_part`”字段。“`sb_part`”字段对应于与将块分割成子块并对经分割的子块进行编码的方法有关的信息。这里，“`sb_part`”的意义根据“`bgmc_mode`”字段的值而改变。

例如，当“`bgmc_mode=0`”时，即当应用 Rice 代码时，“`sb_part=0`”表示该块并不被分割成子块。或者，“`sb_part=1`”表示以 1:4 子块分割比分割该块。或者，当“`bgmc_mode=1`”时，即当采用 BGMC 代码时，“`sb_part=0`”表示以 1:4 子块分割比分割该块。或者，“`sb_part=1`”表示以 1:2:4:8 子块分割比分割该块。

与包含在配置句法（表 6）中的信息相对应的每个块的 `block_data` 句法（表 8）包括 0 比特至 2 比特的可变“`ec_sub`”字段。更具体地，“`ec_sub`”字段指示存在于实际相应块中的子块的数目。这里，“`ec_sub`”字段的意义根据配置句法（表 6）内的“`bgmc_mode`”字段+“`sb_part`”字段的值而变化。

例如，“`bgmc_mode+sb_part=0`”表示 Rice 代码并不配置该子块。这里，“`ec_sub`”字段是个 0 比特字段，这表示不包含任何信息。

除此之外，“`bgmc_mode+sb_part=1`”表示使用了 Rice 代码或 BGMC 代码来以 1:4 的比率将该块分割成若干子块。这里，只有 1 比特被指派给“`ec_sub`”字段。例如，“`ec_sub=0`”指示一个子块（即该块没有分割成多个子块），而“`ec_sub=1`”指示配置了 4 个子块。

此外，“`bgmc_mode+sb_part=2`”表示使用了 BGMC 代码来以 1:2:4:8 的比率将该块分割成若干子块。这里，2 比特被指派给“`ec_sub`”字段。例如，“`ec_sub=00`”指示一个子块（即该块没有分割成多个子块），而“`ec_sub=01`”指示 2 个子块。另外，“`ec_sub=10`”指示 4 个子块，而“`ec_sub=11`”指示 8 个子块。

如上所述定义在每个块内的子块使用差分编码方法由第二熵编码部件 180 进行编码。下面描述使用 Rice 代码的一个示例。对于残差值的每个块，或者可使用同一 Rice 代码编码所有值，或者如果在配置语法中 `b_part` 字段被设置，则该块可被分割成四个子块，每个编码的子块具有一不同的 Rice 代码。

在后一种情况下，块数据句法（表 8）中的“ec_sub”字段指示是使用一个块还是四个块。

尽管第一子块的参数 $s[i=0]$ 或者用 4 比特（分辨率 ≤ 16 比特）或者用 5 比特（分辨率 > 16 比特）直接发送，但仅发送下列参数 $s[i>0]$ 的差分（ $s[i] - s[i-1]$ ）。这些差分还使用适当选择的 Rice 代码再行编码。在这种情况下，差分使用的 Rice 代码参数具有值“0”。

句法

根据本发明的实施方式，包含在音频位流中的各种信息的句法示出于下表中。表 6 示出音频无损编码的配置句法。这种配置句法可形成周期性地置于比特流中的头部，可形成每个帧的帧头等。表 7 示出一种帧—数据句法，而表 8 示出一种块—数据句法。

表 6：配置句法

句法	比特
ALSSpecificConfig() {	
samp_freq;	32
samples;	32
channels;	16
file_type;	3
resolution;	3
floating;	1
msb_first;	1
frame_length;	16
random_access;	8
ra_flag;	2
adapt_order;	1
coef_table;	2
long_term_prediction;	1
max_order;	10
block_switching;	2
bgmc_mode;	1
sb_part;	1
joint_stereo;	1
mc_coding;	1
chan_config;	1
chan_sort;	1
crc_enabled;	1
RLSLMS	1
(reserved)	6
if (chan_config) {	
chan_config_info;	16
}	
if (chan_sort) {	
for (c = 0; c < channels; c++)	
chan_pos[c];	8
}	
header_size;	16
trailer_size;	16
orig_header[];	header size * 8
orig_trailer[];	trailer size * 8
if (crc_enabled) {	
crc;	32
}	
if ((ra_flag == 2) && (random_access > 0)) {	
for (f = 0; f < (samples - 1 /	
frame_length) + 1; f++) {	

<pre> ra_unit_size } } </pre>	32
---	----

表 7: Frame_data (帧_数据) 句法

句法	比特
<pre> frame_data() { if ((ra_flag == 1) && (frame_id % random_access == 0)) { ra_unit_size } if (mc_coding && joint_stereo) { js_switch; byte_align; } if (!mc_coding js_switch) { for (c = 0; c < channels; c++) { if (block_switching) { bs_info; } if (independent_bs) { for (b = 0; b < blocks; b++) { block_data(c); } } else{ for (b = 0; b < blocks; b++) { block_data(c); block_data(c+1); } c++; } } } else{ if (block_switching) { bs_info; } for (b = 0; b < blocks; b++) { for (c = 0; c < channels; c++) { block_data(c); channel_data(c); } } } } </pre>	<p>32</p> <p>1</p> <p>8,16, 32</p> <p>8,16, 32</p>

<pre> } if (floating) { num_bytes_diff_float; diff_float_data(); } </pre>	32
---	----

表 8: Block_data (块_数据) 句法

句法	比特
<pre> block_data() { block_type; if (block_type == 0) { const_block; js_block; (reserved) if (const_block == 1) { { if (resolution == 8) { const_val; } else if (resolution == 16) { const_val; } else if (resolution == 24) { const_val; } else { const_val; } } } } else { js_block; if ((bgmc_mode == 0) && (sb_part == 0) { sub_blocks = 1; } else if ((bgmc_mode == 1) && (sb_part ==1) { ec_sub; sub_blocks = 1 << ec_sub; } else { ec_sub; sub_blocks = (ec_sub == 1) ? 4 : 1; } } } </pre>	<p>1</p> <p>1</p> <p>1</p> <p>5</p> <p>8</p> <p>16</p> <p>24</p> <p>32</p> <p>1</p> <p>2</p> <p>1</p>

<pre> if (bgmc_mode == 0) { for (k = 0; k < sub_blocks; k++) { s[k]; } } else { for (k = 0; k < sub_blocks; k++) { s[k],sx[k]; } } sb_length = block_length / sub_blocks; shift_lsbs; if (shift_lsbs == 1) { shift_pos; } if (!RLSMLS) { if (adapt_order == 1) { opt_order; } for (p = 0; p < opt_order; p++) { quant_cof[p]; } } </pre>	<pre> varie s varie s 1 4 1...10 varie s </pre>
---	---

压缩结果

下面，将无损音频编解码器与用于无损音频压缩的最流行的程序进行比较：开放源编解码器 **FLAC** 和 **Monkey 氏音频(MAC 3.97)**。这里，开放式源代码编解码器 **FLAC** 使用前向自适应预测，而 **Monkey 氏音频 (MAC 3.97)** 是作为压缩方面的当前技术发展水平的算法使用的后向自适应编解码器。这两种编解码器均在有提供最大压缩的选项（即 `flac-8` 和 `mac-c4000`）的情况下运行。编码器的结果是针对中等压缩等级（其预测阶数限制于 `K_60`）以及最大压缩等级（`K_1023`）来确定的，两者均具有 500ms 的随机存取。测试是在有 1024 MB 内存的 1.7GHz 奔腾-M 系统上进行的。测试包括采样率为 48、96 和 192 kHz、分辨率为 16 和 24 比特的将近 1 GB 的立体声波形数据。

压缩率

下面，压缩率被定义为：

$$C = [(\text{压缩的文件大小}) / (\text{原始文件大小})] * 100\%$$

其中越小的值指示越好的压缩。所检查的音频格式的结果示于表 9（FLAC 编解码器不支持 192kHz 的素材）。

表 9：不同音频格式（kHz/比特）的平均压缩率比较

格式	FLAC	MAC	ALS 中值	ALS 最大值
48/16	48.6	45.3	45.5	44.7
48/24	68.4	63.2	63.3	62.7
96/24	56.7	48.1	46.5	46.2
192/24	—	39.1	37.7	37.6
累计	—	48.9	48.3	47.8

这些结果显示，最高等级的 ALS 在所有格式上性能都胜过 FLAC 和 Monkey 氏音频，但对于高清晰度素材（即，96 kHz/24 比特及以上）尤甚。即使在中间等级，ALS 也提供最好的总压缩性。

复杂度

不同编解码器的复杂度强烈地取决于实际实现，尤其是编码器的实现。如上所述，本发明的音频信号编码器仍在发展之中。因此，我们将我们的分析限于解码器——简单的 C 语言代码实现而不作进一步的优化。压缩的数据由当前最佳的编码器实现生成。图 10 中示出了对在不同复杂度等级上编码的各种音频格式进行实时解码所用的平均 CPU 负荷。即使是对于最大复杂度，解码器的 CPU 负荷也只在 20-25%左右，这进而表示基于文件的解码比实时解码快至少 4—5 倍。

表 10：根据音频格式（kHz/比特）和 ALS 编码器复杂度的平均 CPU 负荷（在 1.7GHz 奔腾-M 上的百分比）

格式	ALS 低	ALS 中值	ALS 最大值
48/16	1.6	4.9	18.7
48/24	1.8	5.8	19.6
96/24	3.6	12.0	23.8
192/24	6.7	22.8	26.7

编解码器被设计成可提供大范围的复杂度等级。尽管最大等级以最慢编码

和解码速度为代价实现最高压缩，但较快的中间等级仅些微地降低压缩性，解码的复杂度却显著地低于最大等级（即对于 48 kHz 素材将近 5% 的 CPU 负荷）。使用低复杂度等级（即 K_15, Rice 编码）相比中间等级仅使压缩性降低 1-1.5%，但解码器复杂度进一步降低 3 倍（即对于 48 kHz 的素材而言低于 2% 的 CPU 负荷）。因而，音频数据甚至可以在计算能力很低的硬件上完成解码。

尽管编码器复杂度既会因较高的最大阶数也会因更复杂的块切换算法而增加（根据实施方式）而增大，但解码器会受到较高平均预测阶数的影响。

前面的实施方式（例如分级块切换）和优点仅为示例性的，不应被解释为是对所附权利要求书的限制。本领域技术人员会明白，上述原理可应用于其它装置和方法。许多选择、修改和变化对本领域内技术人员而言是显而易见的。

工业应用性

本领域内技术人员可以理解，可对本发明作出各种修改和变化而不脱离本发明的精神或范围。例如，本发明的诸方面和实施方式很容易在如有损音频信号编解码器等的另一种音频信号编解码器中采用。因此，本发明旨在涵盖本发明的所有这些修改和变化。

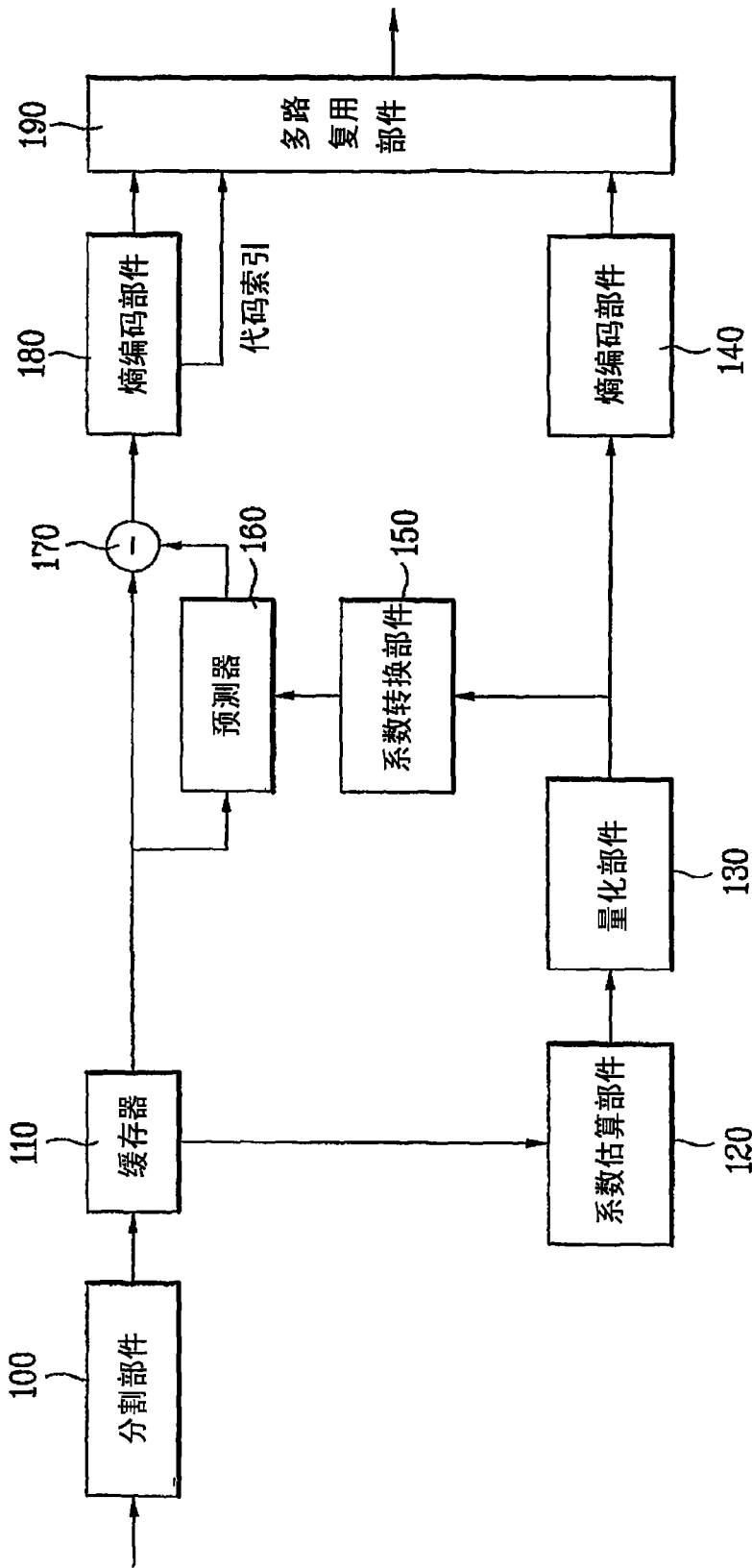
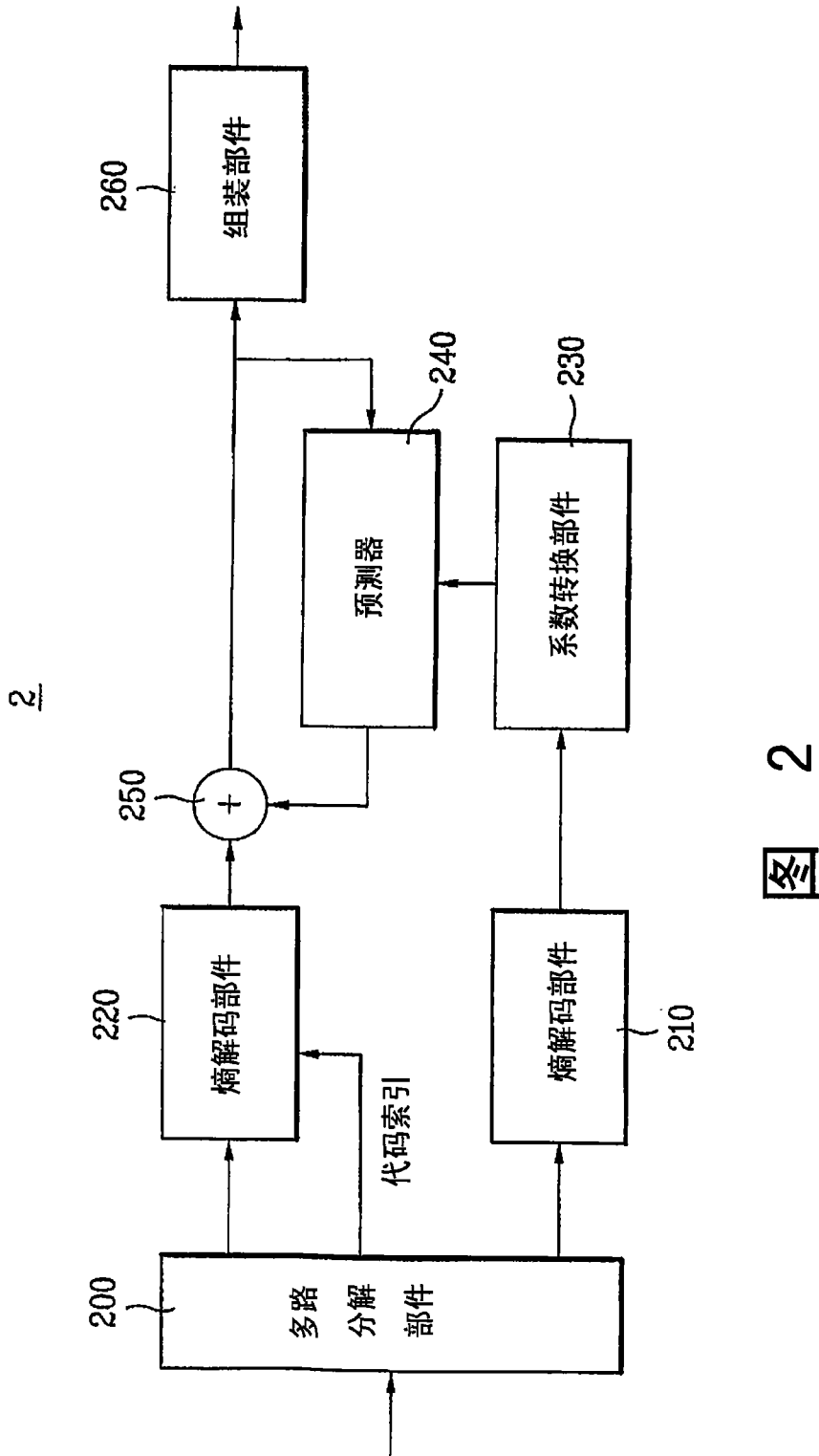


图 1



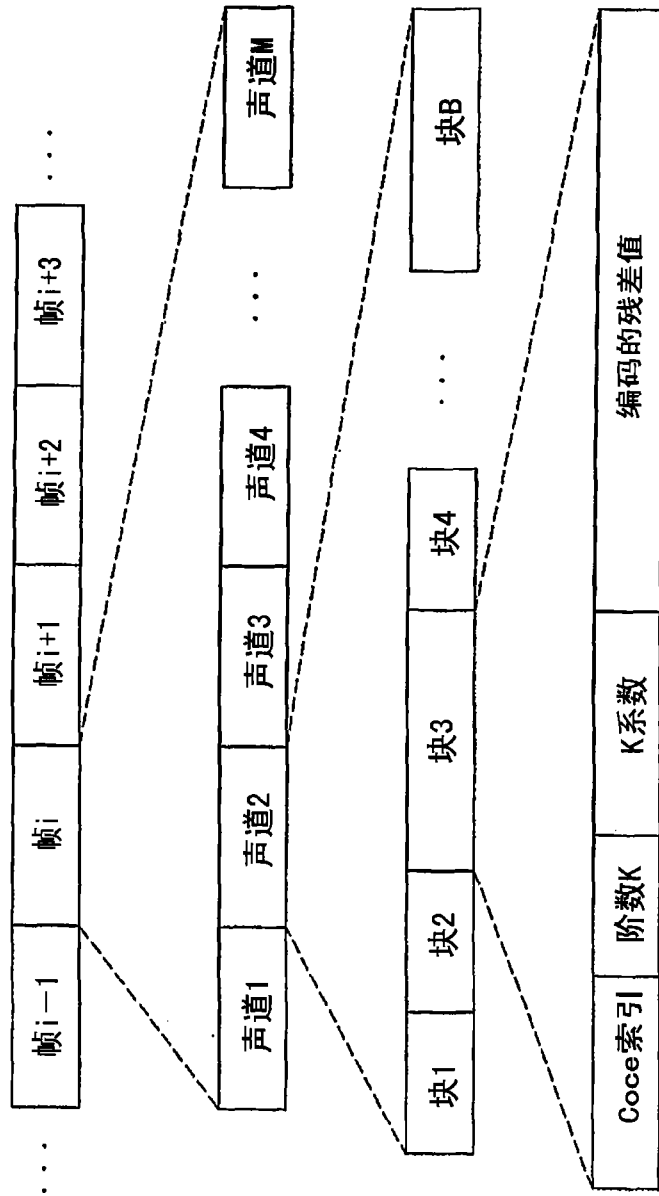


图 3

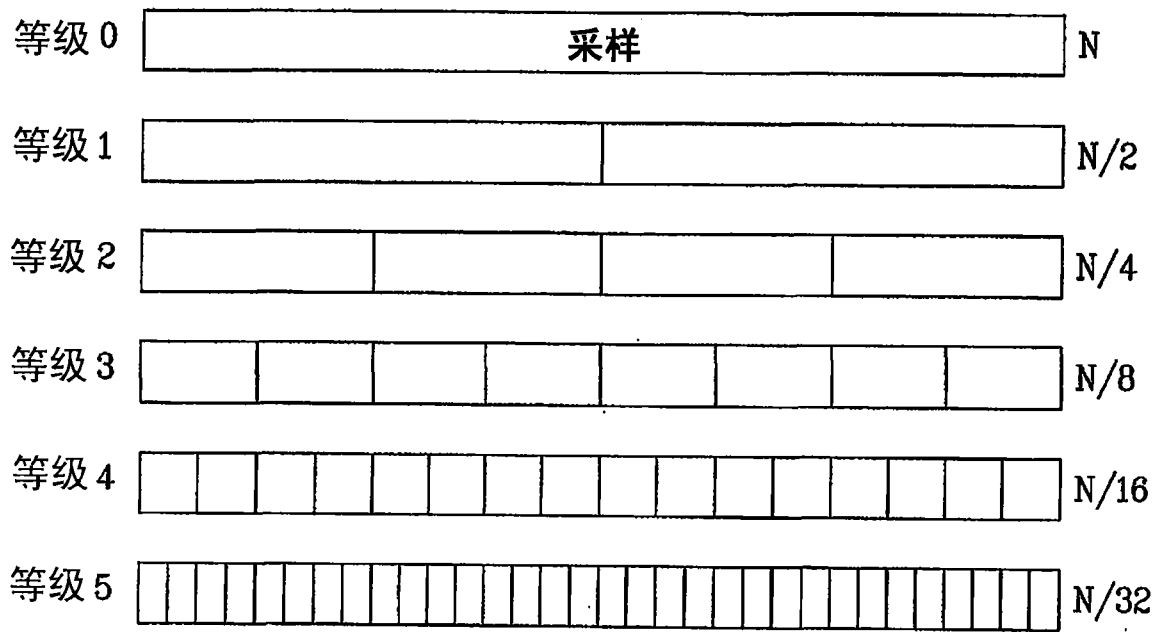


图 4

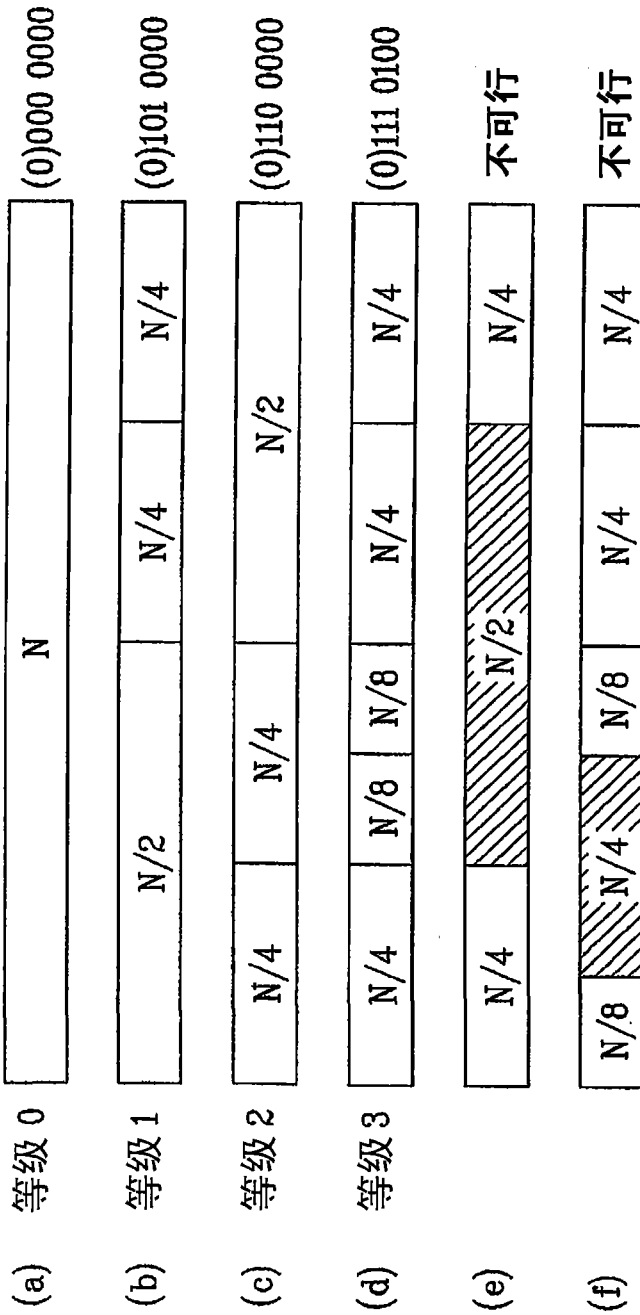


图 5

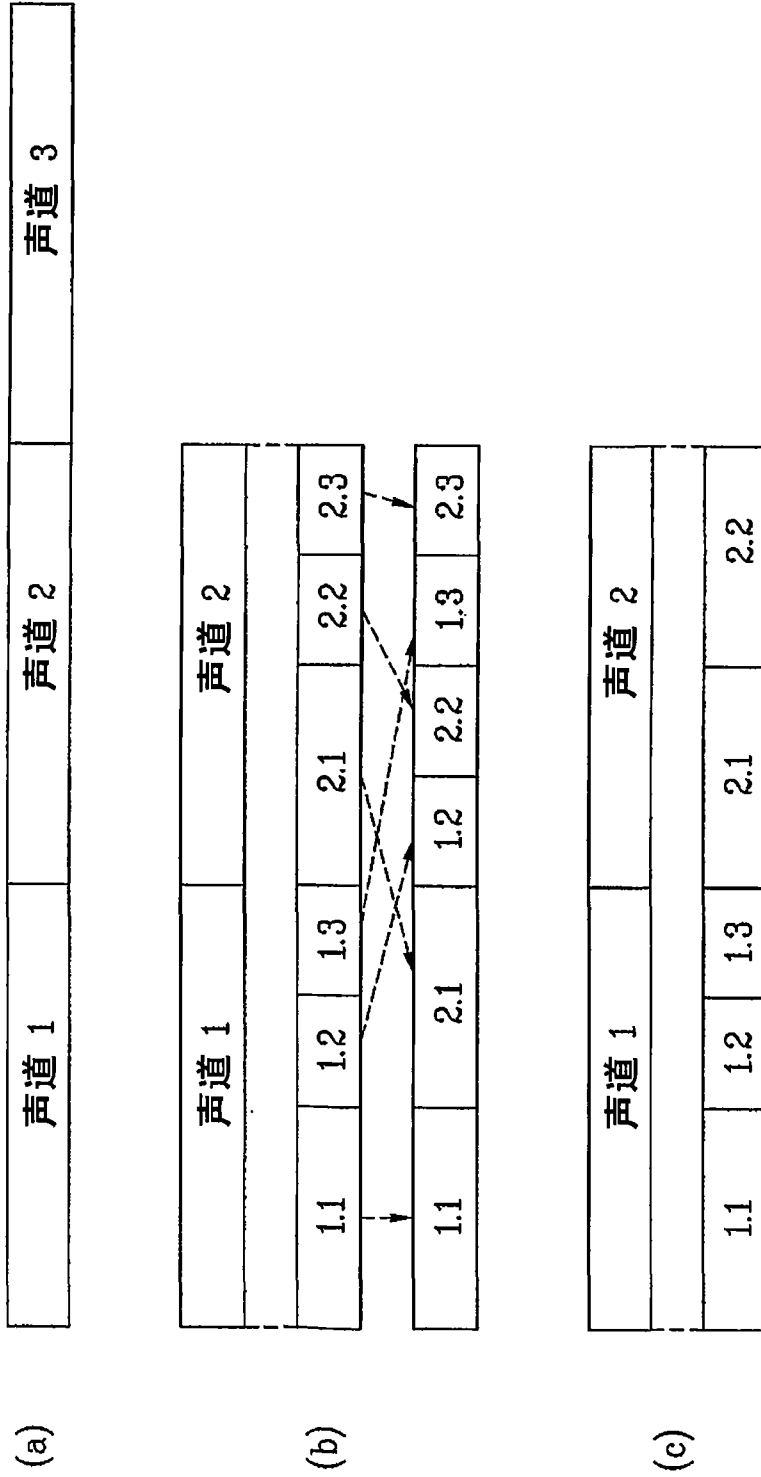


图 6