[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
H04N 5/268
H04N 7/24

[12] 发明专利说明书

[21] ZL 专利号 98801913.2

[45] 授权公告日 2004 年 5 月 5 日

[11] 授权公告号 CN 1148955C

[22] 申请日 1998.10.5 [21] 申请号 98801913.2
[30] 优先权
[32] 1997.10. 3 [33] JP [31] 271275/1997
[86] 国际申请 PCT/JP1998/004497 1998.10.5
[87] 国际公布 WO99/018720 日 1999.4.15
[85] 进入国家阶段日期 1999.7.19
[71] 专利权人 索尼公司
地址 日本东京都
[72] 发明人 田原胜己 安田干太 根岸慎治
审查员 吴黄飞

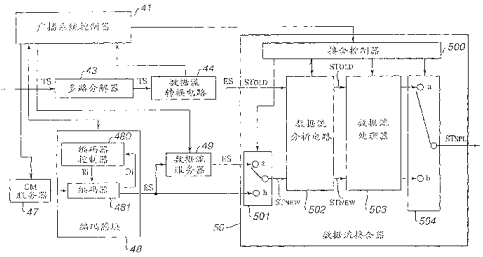
[74] 专利代理机构 北京市柳沈律师事务所
代理人 马 莹

权利要求书 4 页 说明书 41 页 附图 20 页

[54] 发明名称 编码数据流接合设备和方法及编码数据流生成设备和方法

[57] 摘要

一种数据流转换电路，将从主电台发送的原始编码数据流的数据流类型转换成基本数据流。转换后的原始编码数据流提供给数据流接合器的数据流分析电路。数据流分析电路分析所接收的原始编码数据流的语法，以从该原始编码流中提取数据单元，如表示 VBC 缓冲器的初始状态的数据“vbv_delay”、表示在解码是否时生成重复场的标志“repeat_first_field”以及表示第一场是顶场还是底场的标志“top_field_first”，然后将这些数据单元提供给接合控制器，接合控制器又将这些数据单元提供给数据流处理器，数据流处理器根据该数据单元重写替代编码数据流的数据单元。



ISSN1008-4274

1. 一种编码数据流接合设备，用以在接合点接合第一编码数据流和第二编码数据流，该设备包括：

数据流分析装置，用于分析所述第一编码数据流的语法，从而提取所述第一编码数据流的编码参数；及

接合装置，用于根据由所述数据流分析装置得到的所述第一编码数据流的编码参数来改变所述第二编码数据流的编码参数，以便使所述第一编码数据流和所述第二编码数据流在所述接合点无缝连接，然后接合所述第一编码数据流和带有改变后的编码参数的所述第二编码数据流。

2. 如权利要求 1 所述的编码数据流接合设备，其中由所述数据流分析装置所提取的所述编码参数是表示 VBV 缓冲器的初始状态的数据 vbv_delay。

3. 如权利要求 2 所述的编码数据流接合设备，其中所述接合装置将所述第二编码数据流中的接合点之后的第一个画面的 vbv_delay 值重写成所述第一编码数据流中的接合点之后的第一个画面的 vbv_delay 值。

4. 如权利要求 3 所述的编码数据流接合设备，其中所述接合装置将填充字节插入所述第二编码数据流中，以便使所述第二编码数据流中的所述接合点之后的第一个画面的生成比特量变为对应于重写的 vbv_delay 值的比特量。

5. 如权利要求 4 所述的编码数据流接合设备，其中所述接合装置根据所述第一编码数据流中的接合点之后的第一个画面的生成比特量，和所述第二编码数据流中的接合点之后的第一个画面的生成比特量，计算所述填充字节的数据量。

6. 如权利要求 5 所述的编码数据流接合设备，其中所述接合装置根据所述第一编码数据流的接合点的 VBV 缓冲器的数据占有量和所述第二编码数据流的接合点的 VBV 缓冲器的数据占有量，计算所述填充字节的数据量。

7. 如权利要求 1 所述的编码数据流接合设备，其中由所述数据流分析装置提取的所述编码参数是表示在解码时是否生成重复场的标志 repeat_first_field。

8. 如权利要求 7 所述的编码数据流接合设备，其中所述接合装置改变所述第一编码数据流中的接合点之前的画面的 repeat_first_field 值，或改变

所述第二编码数据流中的接合点之前的画面的 repeat_first_field 值，以便使所述第一编码数据流中的接合点之前的画面的帧结构和所述第二编码数据流中的接合点之前的画面的帧结构变得相互一致。

9. 如权利要求 1 所述的编码数据流接合设备，其中由所述数据流分析装置提取的所述编码参数是表示第一场是顶场还是底场的标志 top_field_first。

10. 如权利要求 9 所述的编码数据流接合设备，其中所述接合装置改变所述第一编码数据流中的接合点之前的画面的 top_field_first 值，或改变所述第二编码数据流中的接合点之前的画面的 top_field_first 值，以便使所述第一编码数据流的接合点之前的画面的帧结构和所述第二编码数据流的接合点之前的画面的帧结构变得相互一致。

11. 一种编码数据流接合方法，用于在接合点接合第一编码数据流和第二编码数据流，该方法包括：

数据流分析步骤，用于分析所述第一编码数据流的语法，从而提取所述第一编码数据流的编码参数；以及

接合步骤，用于根据在所述数据流分析步骤得到的所述第一编码数据流的编码参数来改变所述第二编码数据流的编码参数，以便使所述第一编码数据流和所述第二编码数据流在所述接合点无缝连接，然后接合所述第一编码数据流和带有改变后的编码参数的所述第二编码数据流。

12. 如权利要求 11 所述的编码数据流接合方法，其中在所述数据流分析步骤提取的所述编码参数是表示 VBV 缓冲器的初始状态的数据 vbv_delay。

13. 如权利要求 12 所述的编码数据流接合方法，其中在所述接合步骤，将所述第二编码数据流中的接合点之后的第一个画面的 vbv_delay 值重写成所述第一编码数据流中的接合点之后的第一个画面的 vbv_delay 值。

14. 如权利要求 13 所述的编码数据流接合方法，其中在所述接合步骤，将填充字节插入所述第二编码数据流中，以便使所述第二编码数据流中的所述接合点之后的第一个画面的生成比特量变为对应于重写的 vbv_delay 值的比特量。

15. 如权利要求 14 所述的编码数据流接合方法，其中在所述接合步骤，根据所述第一编码数据流中的接合点之后的第一个画面的生成比特量，和所

述第二编码数据流中的接合点之后的第一个画面的生成比特量，计算所述填充字节的数据量。

16. 如权利要求 15 所述的编码数据流接合方法，其中在所述接合步骤，根据所述第一编码数据流的接合点的 VBV 缓冲器的数据占有量和所述第二编码数据流的接合点的 VBV 缓冲器的数据占有量，计算所述填充字节的数据量。

17. 如权利要求 11 所述的编码数据流接合方法，其中在所述数据流分析步骤提取的所述编码参数是表示在解码时是否生成重复场的标志 repeat_first_field。

18. 如权利要求 17 所述的编码数据流接合方法，其中在所述接合步骤，改变所述第一编码数据流中的接合点之前的画面的 repeat_first_field 值，或改变所述第二编码数据流中的接合点之前的画面的 repeat_first_field 值，以便使所述第一编码数据流中的接合点之前的画面的帧结构和所述第二编码数据流中的接合点之前的画面的帧结构变得相互一致。

19. 如权利要求 11 所述的编码数据流接合方法，其中在所述数据流分析步骤提取的所述编码参数是表示第一场是顶场还是底场的标志 top_field_first。

20. 如权利要求 19 所述的编码数据流接合方法，其中在所述接合步骤，改变所述第一编码数据流中的接合点之前的画面的 top_field_first 值，或改变所述第二编码数据流中的接合点之前的画面的 top_field_first 值，以便使所述第一编码数据流的接合点之前的画面的帧结构和所述第二编码数据流的接合点之前的画面的帧结构变得相互一致。

21. 一种编码数据流生成设备，包括：

编码数据流接合设备，用以在接合点接合第一编码数据流和第二编码数据流，该编码数据流接合设备包括：数据流分析装置，用于分析所述第一编码数据流的语法，从而提取所述第一编码数据流的编码参数；及接合装置，用于根据由所述数据流分析装置得到的所述第一编码数据流的编码参数来改变所述第二编码数据流的编码参数，以便使所述第一编码数据流和所述第二编码数据流在所述接合点无缝连接，然后接合所述第一编码数据流和带有改变后的编码参数的所述第二编码数据流；

数据流转换器，用于接收从所述编码数据流接合设备输出的接合的编码

数据流，并且将该接合的编码数据流转换成传输数据流。

22. 一种编码数据流生成方法，包括步骤：

编码数据流接合步骤，用于在接合点接合第一编码数据流和第二编码数据流，该编码数据流接合步骤包括：数据流分析步骤，用于分析所述第一编码数据流的语法，从而提取所述第一编码数据流的编码参数；以及接合步骤，用于根据在所述数据流分析步骤得到的所述第一编码数据流的编码参数来改变所述第二编码数据流的编码参数，以便使所述第一编码数据流和所述第二编码数据流在所述接合点无缝连接，然后接合所述第一编码数据流和带有改变后的编码参数的所述第二编码数据流；

数据流转换步骤，用于接收由所述编码数据流接合步骤输出的接合的编码数据流，并且将该接合的编码数据流转换成传输数据流。

编码数据流接合设备和方法及编码数据流生成设备和方法

技术领域

本发明涉及用在数字广播系统中的编码数据流接合设备(splicing device)、编码数据流接合方法、编码数据流生成设备和编码数据流生成方法。尤其涉及用于通过接合一个数据流级别(a stream level)的两个编码数据流而生成无缝接合的数据流(seamless spliced stream)的编码数据流接合设备、编码数据流接合方法、编码数据流生成设备和编码数据流生成方法。

背景技术

图1描述了当前的电视广播系统。在当前的电视广播系统中，将电视节目分配给各个家庭的广播电台包括制作全国性规模的电视节目的关键电台(或主电台)S_k，及许多附属于关键电台用以制作独特的地方电视节目的地方电台(分支)S_A、S_B和S_C。关键电台S_K用于制作通用全国的电视节目和将制作的电视节目发送给地方电台。地方电台是用于向本地区域的家庭分配从关键电台通过电台间的发送而送来的原始电视节目和通过将原始电视节目的一部分编辑成独特的地方形式而制作成的电视节目的广播电台。例如，如图1所示，地方电台S_A是用于制作要发送给广播区域E_A的家庭的电视节目的电台。地方电台S_B是用于制作要发送给广播区域E_B的家庭的电视节目的电台。地方电台S_C是用于制作要发送给广播区域E_C的家庭的电视节目的电台。例如，在每个地方电台执行的编辑过程是将独特的地方天气预报节目插入到从关键电台发来的新闻节目中的过程，或将地方商业广告插入到诸如电影或戏剧的节目中的过程。

图2A到2C描述了在每个地方电台的编辑过程。图2A表示在关键电台制作的原始电视节目PG_{OLD}。图2B表示在地方电台制作的用于地方观众的替代电视节目PG_{NEW}。图2C表示在地方电台编辑的电视节目PG_{EDIT}。在图2A到2C中所示的编辑过程的是用在地方电台制作的用于地方观众的商业广告CM1'、节目3'和商业广告CM3'来替代从关键电台发来的原始电视节目

的商业广告CM1、节目3和商业广告CM3的例子。这种在地方电台的编辑过程的结果是制作成适合于地方观众的电视节目，其中在关键电台制作的电视节目(即，节目1、节目2、CM2和节目4)和在地方电台制作的电视节目(即，商业广告CM1'、节目3'和CM3')共存。

因为当前电视广播系统使用模拟广播将模拟基带电视信号分配给每个家庭，近年来开始尝试用下一代使用数字技术的广播系统来替代模拟广播系统。数字广播系统是通过使用诸如MPEG2(Moving Picture Experts Group Phase 2，运动画面专家组第二阶段)的压缩编码技术来压缩编码视频和音频数据以及通过地面波或卫星波将编码数据流发送到每个家庭和其它电台的系统。具体讲，从作为数字广播系统提出的广播技术中，欧洲提出的作为下一广播系统的DVB(数字视频广播)标准是最具影响力的技术。该DVB标准已成为事实上的标准。

参照图3，现将描述通过使用MPEG标准从发送端系统将含有视频数据和音频数据的节目发送到接收端系统的典型的数字发送系统。

在该典型的数字发送系统中，发送端系统10有MPEG视频编码器11、MPEG音频编码器12和多路复用器(MUX)13。接收端系统20有多路分解器(DMUX)21、MPEG视频解码器22和MPEG音频解码器23。

MPEG视频编码器11基于MPEG标准对基带源视频数据V编码，并将编码数据流作为视频单元流ES输出。MPEG音频编码器基于MPEG标准对基带原始音频数据A编码，并将编码数据流作为音频单元流ES输出。多路复用器13接收来自MPEG视频编码器11的视频单元流和来自MPEG音频编码器12的音频单元流。然后，多路复用器13将该数据流转换成传输数据流分组的形式，这样就生成了含有视频单元流的传输数据流分组和含有音频单元流的传输数据流分组。多路复用器13对传输数据流多路复用以便使含有视频单元数据流的传输数据流分组和含有音频数据流的传输数据流共存，从而生成要发送给接收系统20的传输数据流。

多路分解器21接收通过传输线发送的传输数据流，并且将传输数据流多路分解成含有视频单元流的传输数据流分组和含有音频单元流的传输数据流分组。然后，多路分解器21从含有视频单元流的传输数据流分组中生成视频单元流，并且从含有音频单元流的传输数据流分组中生成音频单元流。MPEG视频解码器22接收来自多路分解器21的视频单元流，然后基

于 MPEG 标准对该视频单元流解码，从而生成基带视频数据 V。MPEG 音频解码器 23 接收来自多路分解器 21 的音频单元流，并且基于 MPEG 标准对该音频单元流解码，从而生成基带音频数据 A。

在传统的模拟广播系统即将由使用这种数字发送系统的数据广播系统替代的情况下，从关键电台发送到地方电台的电视节目的视频数据是基于 MPEG2 标准进行压缩编码的编码数据流。因此，为了在地方电台执行将关键电台发送来的原始编码数据流的一部分代之以地方电台制作的编码数据流的编辑过程，编码数据流必须在编辑过程执行前被解码以恢复基带视频数据。这是基于以下原因。即，因为按照 MPEG 标准包含在编码数据流中的每个画面的预测方向与前一画面和后一画面的预测方向有关，编码数据流并不能连接在数据流上的任意位置。如果两个编码数据流强行连接，编码数据流的接合处的数据就会不连续且不能正确解码。

因此，要实现参照图 2A 到 2C 所描述的编辑过程，必须执行解码过程，以一次解码从关键电台提供的原始编码数据流和为地方观众制作的编码数据流，从而恢复基带视频信号；还必须执行编辑过程，以编辑两个基带视频数据，从而生成为广播编辑的视频数据；并且还必须执行编码处理，以再次对编辑后的视频数据编码，从而生成编码视频数据流。然而，因为基于 MPEG 标准的编码/解码过程并非 100%可逆的编码/解码过程，所以会出现由于重复解码过程和编码过程而导致的画面质量恶化的问题。

因此，近年来要求有一种在不需对提供的编码数据流解码的情况下就能在编码数据流状态中进行编辑的技术。该技术在编码比特流的级别上连接两个不同的编码比特流，以便生成连接的比特流，这种技术称之为“接合(splicing)”。总之，接合意味着在编码数据流的状态编辑和连接多个数据流。

然而，实现这种接合过程有以下两个问题。

现在将描述第一个问题。

按照用于以上描述的 MPEG 视频解码器 11 和 MPEG 音频解码器 22 的 MPEG 标准，双向预测编码系统将用作编码系统。在这个双向预测编码系统中，执行三种类型的编码，即帧内编码、帧间前向预测编码和双向预测编码。各个类型的编码所得到的画面被称为 I-画面(帧内编码画面)、P-画面(预测编码画面)以及 B-画面(双向预测编码画面)。I-、P-和 B-画面适当组合形成 GOP(画面组)，作为随机存取的单元。通常，I-画面具有最大量的生成代码，

而P-画面具有第二大量的生成代码。B-画面则具有最小量的生成代码。

在为每一画面生成的比特量随 MPEG 的标准而变化的编码方法中，为了准确地由视频解码器 22 对获得的编码比特流(以下简称为数据流)解码以便得到画面，在视频解码器 22 的输入缓冲器中的数据占有量必须被视频编码器 11 所掌握。因此，按照 MPEG 标准，称为“VBV(视频缓冲检验器)缓冲器”的虚拟缓冲器，被假定为对应于视频解码器22 的输入缓冲器的缓冲器，并且规定视频编码器 11 执行编码过程，以便不造成 VBV 缓冲器的故障，即，下溢或上溢。例如，VBV 缓冲器的容量是按照要发送的信号的标准来决定的。在主要轮廓和主要级别(main profile and main level, MP@ML)的标准视频信号的情况下，VBV 缓冲器容量为 1.75M 位。视频编码器 11 控制每个画面生成的比特量以便不会造成 VBV 缓冲器的上溢或下溢。

现在将参照图 4A 到 4C 描述 VBV 缓冲器。

图 4A 表示通过由视频编码器对含有在关键电台制作的节目 1 和商业广告 CM1 的原始电视节目进行编码，得到的原始数据流 ST_{OLD}，以及对应于原始数据流 ST_{OLD} 的 VBV 缓冲器中的数据占有量的轨迹。图 4B 表示由地方电台的视频编码器通过对为地方观众制作的用以替代原始电视节目中的商业广告 CM1 的一部分的商业广告 CM1’进行编码而得到的替代数据流 ST_{NEW}，以及对应于替代数据流 ST_{NEW} 的 VBV 缓冲器中的数据占有量的轨迹。在以下的描述中，因为通过对从关键电台发送到地方电台的原始电视节目进行编码而得到的数据流的一部分被新的数据流所替代，通过对原始电视节目进行编码而得到的原始数据流表示为“ST_{OLD}”，意思是旧的数据流，而重新替代原始数据流 ST_{OLD} 的一部分的替代数据流表示为“ST_{NEW}”。图 4C 表示通过在接合点 SP 接合相对于原始数据流 ST_{OLD} 的替代数据流 ST_{NEW} 而得到的接合数据流 ST_{SPL}，以及对应于接合数据流 ST_{SPL} 的 VBV 缓冲器中的数据占有量的轨迹。

在图 4A 到 4C 中，VBV 缓冲器中的数据占有量的轨迹的右上方部分(斜坡部分)表示发送比特率，而垂直向下部分表示视频解码器从解码器缓冲器中导出的比特量，用于复制每个画面。视频解码器从解码器缓冲器导出比特的时序是按照称为解码时间标记(decoding time stamp, DTS)的信息来指定的。在图 4A 到 4C 中，I、P 和 B 分别代表 I-画面、P-画面和 B-画面。

原始编码数据流 ST_{OLD} 是由关键电台的视频编码器编码的数据流，替代

数据流 ST_{NEW} 是由地方电台的视频编码器编码的数据流。原始编码数据流 ST_{OLD} 和替代数据流 ST_{NEW} 分别被它们各自的视频编码器所编码。因而，由于地方电台的视频编码器是在根本不知道关键电台的视频编码器的 VBV 缓冲器中的数据占有量的轨迹的情况下执行用于独特地生成替代数据流 ST_{NEW} 的编码过程，所以在接合点的 VBV 缓冲器中的原始数据流 ST_{OLD} 的数据占有量 VBV_{OLD} 和在接合点的 VBV 缓冲器中的替代数据流 ST_{NEW} 的数据占有量 VBV_{NEW} 互相不同。

总之，为了防止接合数据流 ST_{SPL} 的接合点 SP 周围的 VBV 缓冲器中的数据占有量的轨迹出现不连续性，VBV 缓冲器中的接合数据流 ST_{SPL} 的替代数据流 ST_{NEW} 的数据占有量的开始级别必须是 VBV 缓冲器中的数据占有量 VBV_{OLD} 的级别。因此，如图 4A 到 4C 所示，如果 VBV 缓冲器中的替代数据流 ST_{NEW} 的数据占有量 VBV_{NEW} 的值小于 VBV 缓冲器中的原始数据流 ST_{OLD} 的数据占有量 VBV_{OLD} 的值，则 VBV 缓冲器在接合数据流 ST_{SPL} 的替代数据流 ST_{NEW} 部分产生上溢。相反，如果 VBV 缓冲器中的替代数据流 ST_{NEW} 的数据占有量 VBV_{NEW} 的值大于 VBV 缓冲器中的原始数据流 ST_{OLD} 的数据占有量 VBV_{OLD} 的值，则 VBV 缓冲器在接合数据流 ST_{SPL} 的替代数据流 ST_{NEW} 的部分产生下溢。

现在将描述第二个问题。

在基于 MPEG 标准编码的数据流的标题中，描述了各种表示编码信息的数据单元和标志。编码数据流是通过使用这些数据单元和标志来解码的。

组成图 2A 到 2C 中所示的原始电视节目的主要部分的节目 1、2、3 和 4 并非一定由通过摄像机等所记录的、具有 29.97Hz(大约 30Hz)的帧速率的 NTSC 系统的电视信号所组成，而可以是由从具有 24Hz(每秒 24 帧)的帧速率的电影材料转变的电视信号所组成。通常，将 24Hz 的电影材料转变成 29.97Hz 的电视信号的处理过程称为“2:3 下拉处理过程”(2:3 pull-down processing)，它包括以预定的顺序将两场的原始材料转变成三场的过程。

图 5 描述了这个 2:3 下拉处理过程。在图 5 中，T1 到 T8 表示具有 24Hz 帧频的电影材料的顶场，B1 到 B8 表示具有 24Hz 的帧频的电影材料的底场。图 5 中的椭圆和三角形表示由顶场和底场所组成的帧结构。

具体讲，在这个 2:3 下拉处理过程中，执行将四个重复场插入到具有 24Hz 的帧频的电影材料(8 个顶场 T1 到 T8 和 8 个底场 B1 到 B8)中的过程。

这四个重复场包括由重复底场 B2 而生成的重复场 B2'，由重复顶场 T4 而生成的重复场 T4'，由重复底场 B6 而生成的重复场 B6'，以及由重复顶场 T8 而生成的重复场 T8'。因此，通过这种 2:3 下拉处理过程，具有 29.97Hz 的帧频的电视信号由具有 24Hz 的帧频的电影材料生成。

在 MPEG 编码器中，由 2:3 下拉处理过程所得到的电视信号不直接由视频编码器所编码，而是在重复场被 2:3 下拉处理的电视信号所除掉后被编码。在图 5 所示的例子中，重复场 B2'、T4'、B6'和 T8'被 2:3 下拉处理的电视信号所除掉。在编码过程前除掉重复场的原因是重复场是在 2:3 下拉处理时插入的冗余场，即便当它们被删掉以提高压缩编码效率时也不会造成任何画面质量的降低。

而且，根据 MPEG 标准，规定了标志“repeat_first_field”，表示是否应该通过重复组成一个帧的任何两个场而生成重复场，该标志是在对编码数据流的解码中描述的。具体讲，在对编码数据流的解码过程中，如果编码数据流中的标志“repeat_first_field”是“1”，MPEG 解码器生成重复场。如果编码数据流中的标志“repeat_first_field”是“0”，MPEG 解码器就不生成重复场。

在图 5 所示的例子中，通过对由顶场 T1 和底场 B1 所组成的帧编码所得到的数据流的“repeat_first_field”是“0”，而通过对由顶场 T2 和底场 B2 组成的帧编码所得到的数据流的“repeat_first_field”是“1”。通过对由顶场 T3 和底场 B3 组成的帧编码所得到的数据流的标志“repeat_first_field”是“0”，而通过对由顶场 T4 和底场 B4 所组成的帧编码所得到的数据流的“repeat_first_field”是“1”。因此，在对由顶场 T2 和底场 B2 组成的帧的数据流解码过程中，生成了重复场 B2'。在对由顶帧 T4 和底帧 B4 组成的帧的数据流解码过程中，生成了重复场 B4'。

另外，按照 MPEG 标准，规定了标志“top_field_first”，表示组成帧的两场的第一场是顶场还是底场，该标志描述在编码数据流中。具体讲，如果“top_field_first”是“1”，表示顶场暂时超前底场的帧结构。如果“top_field_first”是“0”，表示顶场暂时落后底场的帧结构。

在图 5 的例子中，由顶场 T1 和底场 B1 组成的帧的编码数据流的“top_field_first”是“0”，并且由顶场 T2 和底场 B2 组成的帧的编码数据流的“top_field_first”是“0”。由顶场 T3 和底场 B3 组成的帧的编码数据流的标志“top_field_first”是“1”，并且由顶场 T4 和底场 B4 组成的帧的编码数据流的

“top_field_first”是“1”。

参照图 6A 到 6C，现将描述当编码数据流接合时，关于按照 MPEG 标准定义的诸如“top_field_first”和“repeat_first_field”的标志所产生的问题。

图 6A 示出了通过对关键电台制作的原始电视节目编码而得到的原始数据流 ST_{OLD} 的帧结构。图 6B 示出了通过对地方电台为地方观众制作的商业广告 CM’编码而得到的替代数据流 ST_{NEW} 的帧结构。图 6C 示出了通过接合处理过程得到的接合数据流 ST_{SPL} 的帧结构。

在原始数据流 ST_{OLD} 中的节目 1 和节目 2 是通过 2:3 下拉处理过程得到的编码数据流，主要部分的商业广告 CM1 的每个帧是具有“top_field_first”为“0”的帧结构的编码数据流。图 6B 所示的地方商业广告 CM1’是要替代原始电视节目中的商业广告 CM1 的编码数据流，且具有“top_field_first”为“1”的帧结构。图 6C 所示的接合数据流 ST_{SPL} 是通过将替代数据流 ST_{NEW} 接合到节目 1 所表示的原始数据流 ST_{OLD} 后，然后将节目 2 所表示的原始数据流 ST_{OLD} 接合到替代数据流 ST_{NEW} 后所生成的数据流。总之，接合数据流 ST_{SPL} 是通过插入地方商业广告 CM1’以替代原始数据流 ST_{OLD} 的主商业广告 CM1 而得到的数据流。

图 6 所示的在关键电台制作的商业广告 CM1 是其每个帧都具有“top_field_first”为“0”的帧结构的编码数据流。在地方电台制作的商业广告 CM1’是具有“top_field_first”为“1”的帧结构的编码数据流。

如图 6A 到 6B 所示，在商业广告 CM1 的帧结构和要替代商业广告 CM1 的替代商业广告 CM1’的帧结构互相不同的情况下，如果在原始数据流 ST_{OLD} 中的接合点 SP1 将商业广告 CM1’数据流接合到节目 1 的数据流后，则在接合数据流 ST_{SPL} 中生成场隙。场隙意味着底场 B6 在接合点 SP1 从接合数据流 ST_{SPL} 漏掉，这就造成了图 6C 所示的顶场和底场的重复图形中的不连续。

如此在其中生成场隙从而在场图形中造成不连续的编码数据流是与 MPEG 标准不相称并且不能被普通 MPEG 解码器所解码的编码数据流。

另一方面，如图 6A 和 6B 所示，如果节目 2 的数据流在原始数据流 ST_{OLD} 中的接合点被接合到商业广告 CM1’后，在接合数据流 ST_{SPL} 中生成场的副本。如图 6C 所示，场的副本意味着在接合点 SP2 底场 b12 和底场 B12 在相同的显示时间存在。

如此在其中生成场副本从而在场图形中造成不连续的编码数据流是与

MPEG 标准不相称并且不能被普通 MPEG 解码器所解码的编码数据流。

总之，如果接合过程是被简单地执行，则场图形或帧图形就会变得不连续并且不能生成符合 MPEG 标准的接合数据流。

发明内容

本发明的一个目的是提供一种编码数据流接合设备，用以实现无缝接合处理过程，该过程生成在 VBV 缓冲器中的接合数据流的数据占有量的连续轨迹并且在 VBV 缓冲器不产生中断。本发明的另一个目的是提供一种编码数据流接合设备，用以实现无缝接合过程，该过程防止在接合点周围的编码数据流的数据流结构中的中断。

按照本发明，提供了一种编码数据流接合设备，该设备包括：数据流分析装置，用于分析第一编码数据流的语法，从而提取第一编码数据流中的编码参数；以及接合装置，用于根据由数据流分析装置所得到的第一编码数据流的编码参数，改变第二编码数据流的编码参数，以使第一编码数据流和第二编码数据流能够在接合点无缝连接，然后将第一编码数据流和带有改变后的编码参数的第二编码数据流接合。这样，就能实现在 VBV 缓冲器中生成接合数据流的数据占有量的连续轨迹并且不产生 VBV 缓冲器的中断的无缝接合过程。而且，按照本发明，可以实现能够生成具有一致性的无缝数据流，以使接合点周围的编码数据流的数据流结构不中断的接合过程。

按照本发明的另一个方面，还提供了一种利用上面所述的编码数据流接合设备来生成接合的编码数据流的编码数据流生成设备。

按照本发明再一个方面，提供了一种编码数据流接合方法包括：数据流分析步骤，分析第一编码数据流的语法，从而提取第一编码数据流的编码参数；以及接合步骤，用于根据在数据流分析步骤所得到的第一编码数据流的编码参数，改变第二编码数据流的编码参数，从而第一编码数据流和第二编码数据流能够在接合点无缝连接，然后将第一编码数据流和带有改变后的编码参数的第二编码数据流接合。这样，就能实现生成 VBV 缓冲器中的接合数据流的数据占有量的连续轨迹，并且不产生 VBV 缓冲器的中断的无缝接合过程。而且，按照本发明，能够实现能生成具有一致性的无缝数据流，以使接合点周围的编码数据流的数据流结构不中断的无缝接合。

按照本发明的再一个方面，提供了一种编码数据流生成设备，包括：

编码数据流接合设备，用以在接合点接合第一编码数据流和第二编码数据流，该编码数据流接合设备包括：数据流分析装置，用于分析所述第一编码数据流的语法，从而提取所述第一编码数据流的编码参数；及接合装置，用于根据由所述数据流分析装置得到的所述第一编码数据流的编码参数来改变所述第二编码数据流的编码参数，以便使所述第一编码数据流和所述第二编码数据流在所述接合点无缝连接，然后接合所述第一编码数据流和带有改变后的编码参数的所述第二编码数据流；

数据流转换器，用于接收从所述编码数据流接合设备输出的接合的编码数据流，并且将该接合的编码数据流转换成传输数据流。

按照本发明的再一个方面，提供了一种编码数据流生成方法，包括步骤：

编码数据流接合步骤，用于在接合点接合第一编码数据流和第二编码数据流，该编码数据流接合步骤包括：数据流分析步骤，用于分析所述第一编码数据流的语法，从而提取所述第一编码数据流的编码参数；以及接合步骤，用于根据在所述数据流分析步骤得到的所述第一编码数据流的编码参数来改变所述第二编码数据流的编码参数，以便使所述第一编码数据流和所述第二编码数据流在所述接合点无缝连接，然后接合所述第一编码数据流和带有改变后的编码参数的所述第二编码数据流；

数据流转换步骤，用于接收由所述编码数据流接合步骤输出的接合的编码数据流，并且将该接合的编码数据流转换成传输数据流。

附图说明

图 1 描述了典型的电视广播系统。

图 2A 到 2C 描述了在地方电台编辑电视节目的过程。

图 3 描述了使用 MPEG 标准的典型数字发送系统。

图 4A 到 4C 描述了对 VBV 缓冲器的控制。

图 5 描述了 2:3 下拉处理过程。

图 6A 到 6C 描述了在接合编码数据流中产生的问题。

图 7 是表示按照本发明的实施例的包括编码数据流接合设备的数字广播系统的整体结构的方框图。

图 8 是详细解释关键电台 30 的 MPEG 编码器块 35 和地方电台 40 的编码器块的结构的方框图。

图 9A 到 9D 描述了从数据流转换电路中的基本数据流生成传输数据流的过程。

图 10 描述了 MPEG 视频基本数据流的指令序列的语法。

图 11 描述了序列标题的语法(sequence_header)。

图 12 描述了序列扩展的语法(sequence_extension)。

图 13 描述了扩展和用户数据的语法(extension_and_user_data)。

图 14 描述了画面组标题的语法(group_of_picture_header)。

图 15 描述了画面标题的语法(picture_header)。

图 16 描述了画面编码扩展的语法(picture_coding_extension)。

图 17 描述了画面数据的语法(picture_data)。

图 18 描述了传输数据流分组的语法。

图 19 描述了适配场的语法(adaption_field)。

图 20 是用以解释地方电台的简单结构和数据流接合器的结构的方框图。

图 21A 到 21C 描述了关于数据流接合器的 VBV 缓冲器的处理过程。

图 22A 到 22C 描述了关于数据流接合器的“top_field_first”和“repeat_first_field”的第一次处理过程的例子。

图 23A 到 23C 描述了关于数据流接合器的“top_field_first”和

“repeat_first_field”的第二次处理过程的例子。

具体实施方式

图 7 描述了按照本发明的实施例的包括编码数据流接合设备的数字广播系统的结构。

如图 7 所示，数字广播系统一般包括关键电台 30 和附属于该关键电台的地方电台 40 。

关键电台 30 是用于生成普通的电视节目并将其发送到附属地方电台的广播电台。关键电台 30 包括广播系统控制器 31、材料服务器 32、CM 服务器 33、矩阵转换块 34、MPEG 编码器块 35、多路复用器 36 和调制电路 37。

广播系统控制器 31 是用于综合管理和控制所有在广播电台提供的诸如材料服务器 32、CM 服务器 33、矩阵转换块 34 、MPEG 编码器块 35、多路复用器 36 和调制电路 37 等的单元和电路的系统。在该广播系统控制器 31 中，登记一节目编辑表，用以管理所有诸如节目提供公司提供的节目材料以及关键电台本身制作的 CM 材料的所有材料的广播时间。按照节目编辑表，广播系统控制器 31 控制以上描述的单元和电路。例如，节目编辑表由事件信息文件和操作信息文件等组成，在该事件信息文件中记录每小时或每天的广播节目时间表，而在该操作信息文件中记录每 15 秒的广播节目的时间表。

材料服务器 32 是用以存储要作为电视节目的主要部分播出的电影节目、体育节目、娱乐节目和新闻节目的视频数据和音频数据，并且由广播系统控制器 31 在与节目编辑表上的时间表一致的定时处输出指定的节目的服务器。电影节目包括通过如前所述的 2:3 下拉处理过程将具有 24Hz 的帧频的电影材料转换成具有 30Hz 帧频的电视信号所得到的视频数据。作为节目的主要部分从材料服务器 32 输出的视频数据和音频数据提供到矩阵转换块 34。在图 2A 到 2C 所示的例子中，节目 1、节目 2、节目 3 和节目 4 记录在这个材料服务器 32 中。存储在材料服务器 32 中的视频数据和音频数据是没有压缩编码的基带视频数据和音频数据。

CM 服务器 33 是用以存储要在从材料服务器 32 复制的主要部分节目之间插入的商业广告，并且在与节目编辑表上的时间表一致的定时处输出由广播系统控制器 31 指定的商业广告的服务器。作为商业广告从 CM 服务器 33 中输出的视频数据和音频数据提供给矩阵转换块 34。在图 2A 到 2C 所示的

例子中，商业广告CM1、商业广告CM2和商业广告CM3记录在CM服务器33中。存储在CM服务器33中的视频数据和音频数据是没有压缩编码的基带视频数据和音频数据。

矩阵转换块34有用于路由如实况体育广播和新闻节目等的实况节目、从材料服务器32输出的主要节目以及从CM服务器33输出的商业广告节目的矩阵电路。该矩阵转换块34还有一切换电路，用于在与广播系统控制器所决定的节目编辑表上的时间表一致的定时处，连接和切换从材料服务器32供给的主要节目和从CM服务器33供给的商业广告节目。通过使用这样的切换电路来切换主要节目和CM节目，可生成图2A中所示的传输节目PG_{OLD}。

MPEG编码器块35是用于根据MPEG2标准对从矩阵转换块所输出的基带视频数据和音频数据编码的块，并且有多个视频编码器和音频编码器。

多路复用器36对从MPEG编码器块35输出的传输数据流的九条通道进行多路复用，从而生成一个多路传输数据流。因此，在多路传输数据流中，包括九个通道的编码的视频基本数据流的传输数据流分组和包括九个通道的编码的音频基本数据流的传输数据流分组以一种混合的方式共存。

调制电路37执行传输数据流的QPSK调制，并且通过传输线将调制数据发送到地方电台40和家庭61。

现将参照图7描述地方电台40的全面结构。

地方电台40是用以为地方观众编辑从关键电台发送来的普通电视节目的广播电台，并将为地方观众编辑的电视节目播送给每个家庭。地方电台40包括广播系统控制器41、解调电路42、多路分解器43、数据流转换电路44、材料服务器46、CM服务器47、编码器块48、数据流服务器49、数据流接合器50、数据流处理器51、多路复用器52和调制电路53。

广播系统控制器41是用以综合管理和控制在地方电台提供的诸如解调电路42、多路分解器43、数据流转换电路44、材料服务器46、CM服务器47、编码器块48、数据流服务器49、数据流接合器50、数据流处理器51、多路复用器52和调制电路53等的所有单元和电路的系统，类似于关键电台30的广播系统控制器31。在类似于关键电台30的广播系统控制器31的广播系统控制器41中，登记一节目编辑表，用以管理通过将在地方电台制作的节目和CM插入到从关键电台30提供的传输节目中而得到的编辑好的电

视节目的播出时间。按照这个节目编辑表，广播系统控制器 41 控制以上所描述的单元和电路。

解调电路42通过传输线执行从关键电台30发送的传输节目的QPSK解调，从而生成传输数据流。

多路分解器43对从解调电路42输出的传输数据流进行多路分解以便生成九条通道的传输数据流，并且将每个通道的传输数据流输出到数据流转换电路44。总之，多路分解器43执行关键电台30的多路复用器36所执行处理过程的相反过程。

数据流转换电路44是用以将从多路分解器43提供的传输数据流转换成基本数据流的形式的电路。

材料服务器 46 是存储作为电视节目为地方观众广播的娱乐节目、新闻节目等的视频数据和音频数据的服务器。CM 服务器 47 是用以存储要插入到由关键电台 30 提供的主要节目之间的地方商业广告的视频数据和音频数据的服务器。存储在材料服务器46和CM服务器47中的视频数据和音频数据是压缩编码后的基带视频数据和音频数据。

编码器块48是用以将材料服务器46和CM服务器47提供的多个通道的视频数据和多个通道的音频数据进行编码的块，并且具有对应于多个通道的多个视频编码器和多个音频编码器。编码器块48和关键电台的MPEG编码器块之间的不同之处是，地方电台40的编码器块48输出基本数据流，而关键电台30的MPEG编码器块35输出传输数据流。然而，编码器块48的实际功能和处理过程类似于关键电台30的MPEG编码器块35的功能和处理过程。从编码器块 48 输出的多个通道的基本数据流中，三个通道的基本数据流供给数据流服务器49，剩余通道的基本数据流供给数据流接合器50。

数据流服务器 49 接收编码器块提供的三个通道的基本数据流。然后数据流服务器 49 将处于数据流状态的基本数据流记录到可任意读取的记录媒体上，然后在广播系统控制器 41 的控制下从可任意读取的记录媒体复制基本数据流。

数据流接合器50是用于路由编码器块48和数据流服务器49所提供的多个基本数据流，然后将路由后的基本数据流输出到预定的输出线，并且在数据流的级别将关键电台30提供的基本数据流和在地方电台40制作的基本数据流接合的块。接下来将详细描述数据流接合器50的处理过程。

数据流转换电路51是用以接收从数据流接合器50输出的作为接合数据流的基本数据流，并且将该基本数据流转换成传输数据流的电路。

类似于关键电台 30 的多路复用器 36，多路复用器 52 对从数据流转换电路输出的九个通道的传输数据流进行多路复用，从而生成一个多路传输数据流。

调制电路 53 执行传输数据流的 QPSK 调制，并且通过传输线将调制好的数据分配给每个家庭 62。

图 8 是用于详细解释关键电台 30 的 MPEG 编码器块 35 和地方电台 40 的编码器块 48 的结构的方框图。因为关键电台 30 的 MPEG 编码器块 35 和地方电台 40 的编码器块 48 基本上具有相同的结构，将作为一个典型例子来描述关键电台 30 的 MPEG 编码器 35 的结构和功能。

MPEG 编码器块 35 具有：编码器控制器 350，用于以集中方式控制 MPEG 编码器块 35 的所有电路；多个 MPEG 视频编码器 351-1V 到 351-9V，用于对提供的多个通道的视频数据进行编码；以及 MPEG 音频编码器 351-1A 到 351-9A，用于根据 MPEG2 标准对对应于所述视频数据的多个音频数据进行解码。

MPEG 编码器块 35 还具有：数据转换电路 352-1V 到 352-9V，用于将从视频编码器 351-1V 到 351-9V 输出的编码基本数据流(ES)转换成传输数据流；以及数据转换电路 352-1A 到 352-9A，用于将音频编码器 351-1A 到 351-9A 输出的编码基本数据流(ES)转换成传输数据流。MPEG 编码器块 35 还具有：多路复用器 353-1，用于对包括用于每个传输数据流分组的第一通道(1ch)的视频基本数据流的传输数据流和包括第一通道(1ch)的音频基本数据流的传输数据流进行多路复用；多路复用器 353-2，用于多路复用包括用于每个传输数据流分组的第二通道(2ch)的视频基本数据流的传输数据流和包括第二通道(2ch)的音频基本数据流的传输数据流；...;以及多路复用器 353-9，对包括用于每个传输数据流分组的第九通道(9ch)的视频基本数据流的传输数据流和包括第九通道(9ch)的音频基本数据流的传输数据流进行多路复用。

图 8 所示的 MPEG 编码器块 35 有用于对九个通道的传输节目进行编码的结构。然而，通道的数据当然不限于九个而且可以更多。

MPEG 编码器块 35 执行统计多路复用器的控制，以便按照要被编码的

视频数据图形动态地改变每个通道的传输节目的传输速率。根据这项统计多路复用技术，在一个通道的传输节目的画面图形相对简单且仅需要少量的比特来编码，而另一个节目的画面图形相对复杂且需要大量的比特来编码的情况下，对一个通道的画面进行编码的比特被分配给对另一个通道的画面进行编码的比特中，从而实现了传输线的有效传输速率。用于动态改变每个视频编码器的编码速率的方法将在下文中简单描述。

视频编码器351-1V到351-9V首先从在编码过程前作为运动补偿结果所得到的诸如运动补偿残余(motion compensation residual)和AC内(intra-AC)的统计量中，生成难度数据(D1到D9)，每一个难度数据指示对作为编码目标的画面进行编码所要求的比特量。难度数据是表示编码难度的信息。大的难度值表示作为编码目标的画面的图形是复杂的，而小的难度值表示作为编码目标的画面的图形是简单的。该难度数据可以基于诸如在由视频编码器执行编码处理时使用的AC内和运动补偿残余(ME residual)等的统计量来大体估计出。

编码器控制器350接收分别从视频编码器351-1V到351-9V输出的难度数据D1到D9，并且基于难度数据D1到D9分别计算用于视频编码器

351-1V到351-9V的目标比特率R1到R9。具体讲，编码器控制器350能通过使用难度数据D1到D9按比例分配传输线的总的传输率Total_Rate而得到目标比特率R1到R9，如以下公式(1)所示：

$$Ri=(Di/\sum DK)\times Total\_Rate \qquad (1)$$

在公式(1)中，“Ri”代表通道“i”的传输节目的目标比特率，而“Di”代表用于对通道“i”的传输节目进行编码的难度数据。“$\sum$”代表K=1到9通道的难度数据的总和。

编码器控制器350将基于公式(1)计算的目标比特率R1到R9提供给相应的视频编码器351-1V到351-9V。目标比特率R1到R9可以为每个画面或为每个GOP来计算。

视频编码器351-1V到351-9V接收从编码器控制器350提供的目标比特率R1到R9，并且执行对应于目标比特率R1到R9的编码过程。通过这种根据表示要被编码的画面的编码难度的难度数据而动态地改变从每一个视频编码器输出的编码数据流的比特率，可以根据相对于要编码的画面的编码难度来进行分配最佳的比特量，并且防止从视频编码器输出的比特率的总量

上溢于传输线的总的传输率 Total_Rate。

数据流转换电路 352-1V 到 352-9V 以及数据流转换电路 352-1A 到 352-9A 都是用于将基本数据流转换成传输数据流的电路。

参照图 9A 到 9D，将通过一个例子描述从视频基本数据流生成传输数据流的过程，在该例子中，供给的源视频数据通过视频编码器 351-1V 来编码以便生成视频基本数据流，然后被数据流转换电路 352-1V 转换成传输数据流。

图 9A 显示了供给视频编码器 351-V 的源视频数据，而图 9B 显示了从视频编码器 351-1V 输出的视频基本数据流(ES)。图 9C 显示了分组的基本数据流(PES)，而图 9D 显示了传输数据流(TS)。

按照 MPEG2 标准编码的基本数据流的数据量，比如图 9B 中显示的数据流 V1、V2、V3 和 V4，取决于视频帧(I-画面、P-画面或 B-画面)的画面类型和运动补偿的存在/不存在。图 9C 所示的分组的基本数据流(PES)是通过将多个基本数据流分组并在其前端增加了 PES 标题而生成的。例如，PES 标题包括：24 位分组开始代码，表示 PES 分组的开始；8 位流 ID，表示放置在 PES 分组的实际数据部分的数据流数据类型(例如，视频和音频类型)；16 位分组长度，表示随后的数据长度；表示值“10”的代码数据；标志控制部分，其中存储有各种标志信息；8 位 PES 标题长度，表示条件编码部分的数据长度；以及可变长度的条件编码部分，其中存储称为 PTS(Presentation Time Stamp，出现时间标记)的复制输出时间信息和称为 DTS(Decoding Time Stamp，解码时间标记)的解码时的时间管理信息，或存储用以调节数据量的填充字节。

传输数据流(TS)是传输数据流分组的数据列阵，每一个传输数据流由 4 字节的 TS 标题和存储有 184 字节的实际数据的有效载荷部分所组成。为生成传输数据流分组(TS 分组)，首先，为每个 184 字节确定一个 PES 分组的数据流，并且将这 184 个字节的实际数据插入到 TS 分组的有效载荷中。然后，4 个字节的 TS 标题加入到这 184 字节的有效载荷数据中，这样就形成了传输数据流分组。

在下文中，将参照图 10 到 17 描述基本数据流的语法和结构，并将参照图 18 和 19 详细描述传输数据流的语法和结构。

图 10 显示了 MPEG 视频基本数据流的语法。视频编码器块 35 中的视

频编码器 351-1V 到 351-9V 按照图 10 所示的语法生成编码基本数据流。在下文中描述的语法中，函数和条件语句用常规类型表示，而数据单元用粗体类型来表示。数据项描述为名称、比特长度和表示类型和传输序列的助记符。

首先，将描述图 10 中所示的语法中用到的函数。实际上，图 10 中所示的语法是用于从视频解码器一端的传输编码数据流提取预定意义的数据。用于视频编码器一端的语法是从图 10 中的语法中省略 if-子句和 while-子句而得到的语法。

在 video_sequence()中的第一步描述的 next_start_code()函数，是用于搜索比特流中描述的开始代码的函数。按照图 10 中所示语法生成的编码数据流，首先描述了由 sequence_header()函数和 sequence_extension()函数定义的数据单元。sequence_header()函数是用以定义 MPEG 比特流的序列层的标题数据的函数。Sequence_extension()函数是用以定义 MPEG 比特流的序列层的扩展数据的函数。

Do{}while 语句，安排在 sequence_extension()函数之后，Do{}while 语句表示基于 do-子句的函数 in{}所描述的数据单元是在 while-子句为真时所确定的条件下在编码数据流中描述的。在 while-子句中用到的 nextbits()函数用于将比特流中描述的比特或比特阵列与参考数据单元进行比较。在图 10 中显示的语法的例子中，nextbits()函数将比特流中的比特阵列与表示视频序列的末端的 sequence_end_code 进行比较。当比特流中的比特阵列和 sequence_end_code 并不互相一致时，while-子句的条件变为真。因此，安排在 sequence_extension()函数之后的 do{}while 语句，表示在 do-子句中的函数所定义的数据单元是当表示视频序列的末端的 sequence_end_code 没有在比特流中出现时在编码比特流中描述的。

在编码比特流中，由 sequence_extension()函数定义的每个数据单元后面是由 extension_and_user_data(0)定义的数据单元。extension_and_user_data(0)函数是用于以 MPEG 比特流的序列层定义扩展数据和用户数据的函数。

安排在 extension_and_user_data(0)函数之后的 Do{}while 语句，Do{}while 语句函数表示基于 do-子句的函数 in{}而描述的数据单元是在比特流中描述的。在 while-子句中使用的 nextbits()函数是用以确定比特流中出现的比特或比特阵列与 picture_start_code 或 group_start_code 之间的一致性的函数。如果出现在比特流中的比特或比特阵列与 picture_start_code 或

group_start_code 互相一致，则由 while-子句定义的条件变为真。因此，这个 do{}while 语句表示如果 picture_start_code 或 group_start_code 出现在编码比特流中，则在 do-子句中的函数定义的数据单元的代码描述在开始代码之后。

在这个 do-子句的第一步描述的 if-子句表示这样的条件，如，group_start_code 出现在编码比特流中。如果 if-子句的条件为真，则由 group_of_picture_header()函数和 extension_and_user_data(1)函数定义的数据单元依次描述在编码比特流中的 group_start_code 之后。

这个 group_of_picture_header()函数是用于定义 MPEG 编码比特流的 GOP 层的标题数据的函数。Extension_and_user_data(1)函数是用于定义 MPEG 编码比特流的 GOP 层的扩展数据和用户数据的函数。

而且，在该编码比特流中，由 group_of_picture_header()函数和 extension_and_user_data(1)函数定义的数据单元后面紧跟着由 picture_header()函数和 picture_coding_extension()函数定义的数据单元。当然，如果以上描述的 if-子句的条件不是真，则将描述由 group_of_picture_header()函数和 extension_and_user_data(1)函数定义的数据单元。因此，紧接着由 extension_and_user_data(0)函数定义的数据单元，描述了由 picture_header()函数、picture_coding_extension()函数和 extension_and_user_data(2)函数定义的数据单元。

这个 picture_header()函数是用于定义 MPEG 编码比特流的画面层的标题数据的函数。picture_coding_extension()函数是用于定义 MPEG 编码比特流的画面层的第一扩展数据的函数。Extension_and_user_data(2)函数是用于定义 MPEG 编码比特流的画面层的扩展数据和用户数据的函数。由 extension_and_user_data(2)函数定义的用户数据是在画面层中描述的数据并且可为每个画面描述。因此，在本发明中，时间代码信息被描述为由 extension_and_user_data(2)函数定义的用户数据。

在编码比特流中，画面层的用户数据后面紧接着由 picture_data()函数定义的数据单元。Picture_data()函数是用于描述关于限波层(slice layer)和宏块层的数据单元的函数。

紧接着 picture_data()函数描述的 while-子句是用于决定在 while-子句为真时所描述的条件下的下一个 if-子句的条件。用在 while-子句中的 nextbits()函数是用于决定 picture_start_code 或 group_start_code 是否描述在编码比特

流中的函数。如果 picture_start_code 或 group_start_code 是在比特流中描述的，则这个 while-子句所定义的条件变为真。

下一个 if-子句是用于决定 sequence_end_code 是否是在编码比特流中描述的条件句。它表示如果没有描述 sequence_end_code，则描述由 sequence_header()函数和 sequence_extension()函数所定义的数据单元。因为 sequence_end_code 是表示编码视频数据流的序列的末端的编码，所以只要编码数据流没有结束，就将描述 sequence_header()函数和 sequence_extension()函数所定义的数据单元。

由 sequence_header()函数和 sequence_extension()函数所定义的数据单元与视频数据流序列的开端描述的 sequence_header()函数和 sequence_extension()函数所定义的数据单元是完全相同的。在数据流中这样描述相同数据是由于以下原因。即，如果比特流接收装置在数据流的中间点(例如，对应于画面层的比特流部分)开始接收，就不会接收到序列层的数据而且不会对数据流进行解码。描述相同数据的原因是为了防止这种情况。

紧接着由最后一个 sequence_header()函数和 sequence_extension()函数所定义的数据单元，即在数据流的最后部分，描述了表示序列末端的 32-位 sequence_end_code。

现在将详细描述 sequence_header()函数、sequence_extension()函数、extension_and_user_data(0) 函数、group_of_picture_header() 函数和 extension_and_user_data(1)函数。

图 11 描述了 sequence_header()函数的语法。由 sequence_header()函数所定义的数据单元包括 sequence_header_code、sequence_header_present_flag、horizontal_size_value、vertical_size_value、aspect_ratio_information、frame_rate_code、bit_rate_value、marker_bit、vbv_buffer_size_value、constrained_parameter_flag、load_intra_quantizer_matrix、intra_quantizer_matrix、load_non_intra_quantizer_matrix 和 non_intra_quantizer_matrix。

Sequence_header_code 是表示序列层的开始同步代码的数据。Sequence_header_present_flag 是表示 sequence_header 中的数据是合法还是非法的数据。Horizontal_size_value 是由画面的水平像素数的低 12 位组成的数据。Vertical_size_value 是由画面的垂直线数的低 12 位组成的数据。

Aspect_ratio_information 是表示像素的纵横比或显示屏的纵横比的数据。Frame_rate_code 是表示画面的显示周期的数据。Bit_rate_value 是限制生成的比特量的比特率的低 18 位(大约每秒 400bps)的数据。Marker_bit 是为防止开始代码模拟(emulation)而要插入的数据。Vbv_buffer_size_value 是决定用以控制生成的代码量的虚拟缓冲器(视频缓冲器检验器)的大小的值的低 10 位数据。Constrained_parameter_flag 是表示每个参数都在其限制内的数据。Load_intra_quantizer_matrix 是表示存在用于 MB 内(intra MB)的量化矩阵数据的数据。Intra_quantizer_matrix 是表示用于 MB 内的量化矩阵数据值的数据。Load_non_intra_quantizer_matrix 是表示存在用于非 MB 内(non-intra MB)的量化矩阵数据的数据。Non_intra_quantizer_matrix 是表示用于非 MB 内的量化矩阵值的数据。

图 12 描述了 sequence_extension()函数的语法。由 sequence_extension()函数所定义的数据单元包括 extension_start_code、extension_start_code_identifier、sequence_extension_present_flag、profile_and_level_indication、progressive_sequence、chroma_format、horizontal_size_extension、vertical_size_extension、bit_rate_extension、vbv_buffer_size_extension、low_delay、frame_rate_extension_n 和 frame_rate_extension_d。

Extension_start_code 是表示扩展数据的开始同步代码的数据。Extension_start_code_identifier 是表示要被发送的扩展数据的数据。Sequence_extension_present_flag 是表示序列扩展中的数据是合法还是非法的数据。Profile_and_level_indication 是用于指定轮廓和视频数据的级别的数据。Progressive_sequence 是表示视频数据是依次扫描的数据的数据。Chroma_format 是指定视频数据的色差格式的数据。Horizontal_size_extension 是要加入到序列标题的 horizontal_size_value 中的高 2 位数据。Vertical_size_extension 是要加入到序列标题的 vertical_size_value 中的高 2 位数据。Bit_rate_extension 是要加入到序列标题的 bit_rate_value 中的高 12 位数据。Vbv_buffer_size_extension 是要加入到序列标题的 vbv_buffer_size_value 中的高 8 位数据。Low_delay 是表示不含有 B-画面的数据。Frame_rate_extension_n 是用于得到与序列标题的 frame_rate_code 结合的帧频的数据。Frame_rate_extension_d 是用于得到与序列标题的

frame_rate_code 结合的帧频的数据。

图 13 描述了 extension_and_user_data(i) 函数的语法。如果 extension_and_user_data(i)的"i"不是 2，则仅描述由 user_data()函数定义的数据单元，而不描述由 extension_data()函数定义的数据单元。因此，extension_and_user_data(0)函数仅描述由 user_data()函数定义的数据单元。

图 14 描述了 group_of_picture_header() 函数的语法。由 group_of_picture_header()函数定义的数据单元包括 group_start_code、group_of_picture_header_present_flag、time_code、closed_gop 和 broken_link。

Group_start_code 是表示 GOP 层的开始同步代码的数据。Group_of_picture_header_present_flag 是表示 group_of_picture_header 中的数据单元是合法还是非法的数据。Time_code 是表示 GOP 的引导画面的序列的引导端时间的时间编码。Closed_gop 是表示 GOP 中的画面能独立于其它 GOP 而复制的标志数据。Broken_link 是表示 GOP 中的引导 B-画面不能准确地复制用于编辑和作其它类似工作的标志数据。

Extension_and_user_data(1)函数是仅描述由 user_data()函数定义的数据单元的函数，类似于 extension_and_user_data(0)函数。

参照图 15 到 17，现将描述用于描述关于编码数据流的画面层的数据单元的 picture_header() 函数、picture_coding_extension() 函数、extension_and_user_data(2)函数和 picture_data()函数。

图 15 描述了 picture_header()函数的语法。由 picture_header()函数定义的数据单元包括 picture_start_code、temporal_reference、picture_coding_type、vbv_delay、full_pel_forward_vector、forward_f_code、full_pel_backward_vector、backwark_f_code、extra_bit_picture 和 extra_information_picture。

具体讲，picture_start_code 是表示画面层的开始同步代码的数据。Temporal_reference 是在 GOP 的引导端重置的数据，带有表示画面显示顺序的数字。Picture_coding_type 是表示画面类型的数据。

Vbv_delay 是表示 VBV 缓冲器的开始状态的数据，并且为每个画面设置。从发送端系统发送到接收端系统的编码基本数据流的画面是由接收端系统中提供的 VBV 缓冲器来缓冲的。画面是从 VBV 缓冲器导出(读出)的并且在由 DTS(解码时间标记)指定的时间提供给解码器。由 vbv_delay 定义的时

间是解码目标的画面开始从VBV缓冲器导出直到编码目标的画面从VBV缓冲器中导出的时间，即，直到由 DTS 指定的时间。在本发明的编码数据流接合设备中，无缝接合，诸如 VBV 缓冲器中数据占有量不再中断，是通过使用存储在画面标题中的 vbv_dalay 来实现的。接下来将详细描述这种特征。

Full_pel_forward_vector 是表示前向运动矢量的精确度是基于整数单元还是半像素单元的数据。Forward_f_code 是表示前向运动矢量搜索范围的数据。Full_pel_backward_vector 是表示后向运动矢量的精确度是基于整数单元还是半像素单元的数据。Backward_f_code 是表示后向运动矢量搜索范围的数据。Extra_bit_picture 是表示随后的附加信息存在的标志。如果 extra_bit_picture 是“1”，则紧接着为 extra_information_picture。如果 extra_bit_picture 是“0”，则没有数据跟随。Extra_information_picture 是按照标准储备的信息。

图 16 描述了 picture_coding_extension() 函数的语法。由 picture_coding_extensin() 函数定义的数据单元包括 extension_start_code、extension_start_code_identifier、f_code[0][0]、f_code[0][1]、f_code[1][0]、f_code[1][0]、f_code[1][1]、intra_de_precision、picture_structure、top_field_first、frame_predictive_frame_dct、concealment_motion_vectors、q_scale_type、intra_vlc_format、alternate_scan、repeat_first_field、chroma_420_type、progressive_frame、composite_display_flag、v_axis、field_sequence、sub_carrier、burst_amplitude 及 sub_carrier_phase。

extension_start_code 是表示画面层的扩展数据的开始的开始代码。Extension_start_code_identifier 是表示要被发送的扩展数据的编码。F_code[0][0]是表示前向水平运动矢量的搜索范围的数据。F_code[0][1]是表示前向垂直运动矢量的搜索范围的数据。F_code[1][0]是表示后向水平运动矢量的搜索范围的数据。F_code[1][1]是表示后向垂直运动矢量的搜索范围的数据。Intra_dc_precision 是表示 DC 系数的准确性的数据。Picture_structure 是表示画面结构是帧结构还是场结构的数据。在场结构的情况下，它也表示该场是较高场还是较低场。

Top_field_first 是表示在帧结构情况下第一场是顶场还是底场的标志。Frame_predictive_frame_dct 是表示帧模式 DCT 的预测值仅仅是帧结构情况下的帧模式的数据。Concealment_motion_vectors 是表示用于隐藏传输差错的

运动矢量是在宏块内提供的数据。Q_scale_type 是表示使用线性量化器标准还是非线性量化器标准的数据。Intra_vlc_format 是表示是否将另一个二维 VLC 用于宏块内的数据。Alternate_scan 是指示选择之形描述还是交替描述的数据。

Repeat_first_field 是表示是否在解码时生成重复场的标志。在解码过程中，如果 repeat_first_field 为"1"，就生成重复场。如果 repeat_first_field 为"0"，就不生成重复场。Chroma_420_type 是表示在信号格式为 4:2:0 时与下一个 progressive_frame 相同值的数据，而在信号格式不是 4:2:0 时表示 0。Progressive_frame 是表示该画面是否被序列扫描的数据。Composite_display_flag 是表示源信号是否为复合信号的数据。V_axis 是当源信号为 PAL 系统的情况下使用的数据。Field_sequence 是源信号为 PAL 系统的情况下使用的数据。Sub_carrier 是源信号为 PAL 系统的情况下使用的数据。Burst_amplitude 是源信号为 PAL 系统的情况下使用的数据。Sub_carrier_phase 是源信号为 PAL 系统的情况下使用的数据。

Extension_and_user_data(2)函数描述当编码比特流中存在扩展开始代码(extension_start_code)时 extension_data()函数所定义的数据单元，如图 13 所示。然而，如果在比特流中不存在扩展开始代码，在比特流中将不描述由 extension_data()函数定义的数据单元。如果在比特流中存在用户数据开始代码(user_data_start_code)，对 user_data()函数定义的数据单元将紧接着 entension_data()函数定义的数据单元进行描述。

图 17 描述了 picture_data()函数的语法。Picture_data()函数定义的数据单元是由 slice()函数所定义的数据单元。然而，如果在比特流中不存在表示 slice()函数的开始代码的 slice_start_code，比特流中将不描述由 slice()函数定义的数据单元。

Slice()函数是描述关于分层的数据单元的函数。具体讲，该函数据描述诸如 slice_start_code、slice_quantiser_scale_code、intra_slice_flag、intra_slice 、reserved_bits、extra_bit_slice、extra_information_slice 和 extra_bit_slice 以及 macroblock()函数定义的数据单元。

Slice_start_code 是表示 slice()函数定义的数据单元的开始的开始代码。Slice_quantiser_code 是表示为分层中存在的宏块设置的量化步长的数据。然而，如果为每个宏块设置 quantiser_scale_code，将优选使用为每个宏块设置

的 macroblock_quantiser_scale_coder 的数据。Intra_slice_flag 是表示在比特流中是否存在 intra_slice 和 reserved_bits 的标志。Intra_slice 是表示在分层中是否存在非宏块内的数据。如果分层中的任何一个宏块都是非宏块内，intra_slice 是“0”。如果分层中的所有宏块都是非宏块内，intra_slice 是“1”。Reserved_bits 是 7 位数据且值为“0”。Extra_bit_slice 是表示附加信息作为编码数据流存在的标志。如果 extra_information_slice 存在于 extra_bit_slice 之后，该标志设置为“1”。如果不存在附加信息，标志设置为“0”。

Macroblock()函数是用于描述关于宏块层的数据单元的函数。具体讲，它是用于描述诸如 macroblock_escape、macroblock_address_increment、及 macroblock_quantiser_scale_code 的数据单元，以及由 macroblock_mode()函数和 macroblock_vectors(s)函数定义的数据单元。

Macroblock_escape 是表示参考宏块与前一宏块间的水平差值是否不小于 34 的固定比特阵列。如果参考宏块与前一宏块间的水平差值不小于 34，在 macroblock_address_increment 的值上加 33。Macroblock_address_increment 是表示参考宏块与前一宏块之间的水平差值的数据。如果在这个 macro_address_increment 之前存在一个 macroblock_escape，macroblock_address_increment 的值上增加 33 所得到的值变为表示参考宏块与前一宏块之间的实际水平差值的数据。Macroblock_quantiser_scale_code 表示为每个宏块设置的量化步长。在每个分层中，slice_quantiser_scale_code 表示设置的分层的量化步长。然而，如果 macroblock_quantiser_scale_code 是为参考宏块设置的，则选择这个量化步长。

参照图 18 和 19，将详细描述传输数据流分组的结构和传输数据流分组的语法。

传输数据流分组由 4 字节的标题、用于存储各种数据和数据单元的 184 字节的有效载荷部分组成。

传输数据流分组的标题包括各种字段，比如 sync_byte、transport_error_indicator、payload_unit_start_indicator、transport_priority、PID、transport_scrambling_control、adaptation_field_control、continuity_counter 和 adaptation_field。

Sync_byte 是用于从比特流检测同步图形的固定 8 比特段。其值由固定值“01000111”(0x47)所定义。通过检测数据流中的这个同步图形，能够检测

同步。

Transprot_error_indicator 是 1 比特标志。如果该标志设为“1”，则表示在传输数据流分组中至少存在一位的不可校正比特差错。

Payload_unit_start_indicator 是 1 比特标志。它是具有用于诸如视频/音频数据或用以发送节目规范信息(PSI)的传输数据流分组的基本数据的规定意义的数据。如果传输数据流分组的有效载荷包括基本数据，payload_unit_start_indicator 具有以下意思。如果 payload_unit_start_indicator 为“1”，它表示基本数据插入在传输数据流分组的有效载荷的开端。如果 payload_unit_start_indicator 为“0”，基本数据不是插入到传输数据流分组的有效载荷的开端。如果 payload_unit_start_indicator 设置为“1”,它表示仅仅一个 PES 分组以任意传输数据流分组开始。另一方面，如果传输数据流的有效载荷包括 PSI 数据，payload_unit_start_indicator 具有以下意思。如果传输分组发送 PSI 部分的第一字节，payload_unit_start_indicator 变为“1”。如果传输数据流分组不发送 PSI 部分的第一字节，payload_unit_start_indicator 变为“0”。而且，如果传输数据流分组是一个零分组，payload_unit_start_indicator 变为“0”。

Transport_priotity 是 1 比特标识符，表示传输分组的优先权。如果 transport_priority 设为“1”，它表示该传输分组具有优先于具有相同分组标识符 PID 并且 transport_priority 不是“1”的分组的优选权。例如，通过设定 transport_priority 的这个分组标识符，优先权可以给于一个基本数据流中的任意分组。

Transport_scrambling_control 是表示传输数据流分组的加优模式的 2 比特数据。加扰模式是表示存储在有效载荷部分中的数据是否被加扰以及加扰类型的模式。传输数据流分组标题与适应字段都是标准化的，不由加扰键 Ks 加扰。因此，通过 transport_scrambling_control，它能决定存储在传输数据流分组中的有效载荷部分中的数据是否被量化。

Adaptation_field_control 是表示适应字段和/或有效载荷排列在传输数据流分组的分组标题中的 2 比特数据。具体讲，如果在分组标题中排列的只有有效载荷数据，adaptation_field_control 变为“01”。如果在分组标题中排列的只有适应字段，adaptation_field_control 变为“10”。如果适应字段和有效载都排列在分组标题中，adaptation_field_control 变为“11”。

Continuity_counter 是表示具有被连续发送的相同的 PID 的分组的一部分是否丢失或转储(dump)的数据。具体讲，continuity_counter 是为每个具有相同 PID 的传输数据流分组增加的 4 比特字段。然而，当 continuity_counter 计数时，适应字段安排在分组标题中。

Adaptation_field()是用于作为选择将与单独数据流或填充字节有关的附加信息插入的字段。通过适应字段，所有关于单独数据流状态的动态变化的信息能与数据一起进行发送。

图 19 描述了 adaptation_field()的语法。这个 adaptaton_field()包括各种字段，诸如 adaptation_field_length、discontinuity_counter、random_access_indicator、elementary_stream_priority_indicator、OPCR_flag、splicing_point_flag、transport_private_data_flag、adaptation_field_extension_flag、program_clock_reference(PCR)、original_program_clock_reference(OPCR)、splice_countdown、transport_private_data_length、private_data、adaptation_field_extension_length、ltw_flag(legal_time_window_flag)、piecewise_rate_flag 和 seamless_splice_flag。

Adaptation_field_length 是表示在 adaptation_field_length 之后的适应字段的字节数的数据。如果 adaptation_field_control 为“11”，adaptation_field_length 为 0 到 182 比特。如果 adaptation_field_control 为“10”，adaptation_field_length 为 183 比特。同时，如果不存在足以填充传输数据流的有效载荷的基本数据流，则需要用于填充比特的缓冲过程。

Discontinuity_counter 是表示系统时钟基准(SCR)是否在具有相同 PID 的多个分组的中间部分复位并因此变得不连续的数据。如果系统时钟基准不连续，则 discontinuity_counter 为“1”。如果系统时钟基准连续，则 discontinuity_counter 为“0”。系统时钟基准是基准信息，用于在对视频和音频数据进行解码的 MPEG 解码器中，在编码器端想要的定时处，设定解码器端的系统时钟值。

Random_access_indicator 是表示视频序列标题或音频帧的开始的数据。总之，random_access_indicator 是表示在执行对数据单元的随机存取过程中的视频或音频存取点(帧的开始)的数据。

Elementary_stream_priority_indicator 是表示相对于具有相同 PID 的分组，要在传输数据流分组的有效载荷中发送的基本数据流数据的优先权的数

据。例如，如果基本数据流含有内编码的视频数据，则elementary_stream_priority_indicator 设置为“1”。否则，elementary_stream_priority_indicator 设置为“0”。

PCR_flag 是表示适应字段中是否存在 PCR(program_clock_reference)数据的数据。如果在适应字段中存在 PCR 数据，PCR_flag 设置为“1”。如果适应字段中不存在 PCR 数据，PCR_flag 设置为“0”。这个 PCR 数据是用于获取接收装置的解码器中发送数据的解码过程的定时的数据。

OPCR_flag 是表示适应字段中是否存在 OPCR(original_program_clock_reference)数据的数据。如果在适应字段中存在 OPCR 数据，OPCR_flag 设置为“1”。如果在适应字段中不存在 OPCR 数据，OPCR_flag 设置为“0”。OPCR 数据是用于通过接合过程从多个原始传输数据流重构一个传输数据流时使用的数据。OPCR 数据是表示确定的原始传输数据流的 PCR 数据的数据。

Splicing_point_flag 是表示在适应字段中是否存在用于表示传输级别的编辑点(接合点)的 splice_countdown 的数据。如果在适应字段中存在 splice_countdown，则 splice_point_flag 为“1”。如果在适应字段中不存在 splice_countdown，则 splice_point_flag 为“0”。

Transport_private_data_flag 是表示在适应字段中是否存在描述随机用户数据的私有数据的数据。如果在适应字段中存在私有数据，则 transport_private_data_flag 设置为“1”。如果在适应字段中不存在私有数据，则 transport_private_data_flag 设置为“0”。

Adaptation_field_extension_flag 是表示在适应字段中是否存在扩展字段的数据。如果在适应字段中存在扩展字段，则 adaptation_field_extension_flag 设置为“1”。如果在适应字段中不存在扩展字段，则 adaptation_field_extension_flag 设置为“0”。

Program_clock_reference(PCR)是指用于使接收端的时钟状态与发送端的时钟状态同步的基准时钟。PCR 数据存储生成传输分组的时间。PCR 数据由 33 比特的 program_clock_reference_base 和 9 比特的 program_clock_reference_extension 所组成，即，总共 42 比特。系统时钟由 program_clock_reference_extension 从 0 到 299 计数，通过载波从 299 到 0 重置将 1 比特加到 program_clock_reference_base。这样就能够进行 24 小时计数。

Original_program_clock_reference(OPCR)是从确定的传输数据流重构单个节目的传输数据流的数据。如果单个节目传输数据流被完全重构，original_program_clock_reference 被复制到 program_clock_reference。

Splice_countdown 是表示分组的数目上升到编辑(接合过程)可在相对于具有相同 PID 的传输数据流分组的传输数据流分组级别上执行的点的数据。因此，在编辑可执行的接合点的传输数据流分组中，splice_countdown 为“0”。在具有 splice_countdown 为“0”的传输分组中，接合过程可通过使传输数据流分组有效载荷的最后字节成为编码画面的最后字节来执行。

接合过程是用于接合在传输级别的两个不同的基本数据流，从而生成一个新的传输数据流的过程。接合过程可被分为不生成代码中断的无缝接合和生成代码中断的非无缝接合。“不生成代码中断”意味着在最新连接的后续数据流的存取单元的解码时间与接合前旧的数据流的存取单元的解码时间之间不存在矛盾。“生成代码中断”意味着在最新连接的后续数据流的存取单元的解码时间与接合前旧的数据流的存取单元的解码时间之间产生矛盾。

Transport_private_data_length 是表示适应字段中的私有数据的字节数的数据。

Private_data 字段不是特别由标准规定，并且可以描述适应字段中的任意用户数据。

Adaptation_field_extension_length 是表示适应字段中的适应字段扩展的数据长度的数据。

Ltw_flag(legal_time_window_flag)是表示在适应字段中是否存在表示显示窗口补偿值的 ltw_offset 的数据。

Piecewise_rate_flag 是表示在适应字段中是否存在 piecewise_rate 的数据。

Seamless_slice_flag 是表示接合点是正常接合点还是无缝接合点的数据。如果 seamless_splice_flag 为“0”，它表示接合点为正常接合点。如果 seamless_splice_flag 为“1”，它表示接合点为无缝接合点。正常接合点位于 PES 分组的末端。紧接在该接合点之前的接合分组以一访问存取结束，并且具有相同 PID 的传输分组开始于 PES 分组的标题。相反，无缝接合点存在于 PES 分组的中间点。为防止在最新接合的后续数据流的存取单元的解码时间与接合前旧的数据流的存取单元的解码时间之间出现矛盾，一部分旧的数

据流的特征用作新的数据流的特征。

下面将参照图 20 至 23 描述接合处理过程，该过程用于接合从关键电台 30 发送来的数据流 ST_{OLD} 和在地方电台 40 产生的数据流 ST_{NEW}。

在图 20 中，为了阐明图 7 描述的地方电台 40 的控制，仅显示了多个通道中的一个通道而别的通道都被省略。在本发明中，提供了接合过程的三个实施例。现在将依次描述接合过程的第一、第二和第三实施例。

在接合过程的第一实施例中，接合过程是在这样的情况下进行的：在发送节目的编码数据流 ST_{OLD} 从关键电台 30 发送之前，要被最新插入的商业广告 CM'的编码数据流 ST_{NEW} 已经生成。也就是说，提前编码的商业广告 CM1'的数据流被插入到发送节目的商业广告 CM 的编码数据流 ST_{OLD} 中。通常，商业广告重复广播一定的时间。因此，每次广播时都对商业广告的视频数据进行编码是没有效率的。这样，地方商业广告 CM1'的视频数据被编码，而其编码数据流 ST_{NEW} 存储在数据流服务器 49 中。然后，当要被替代的商业广告 CM1 的编码数据流 ST_{OLD} 从关键电台 30 传来时，地方商业广告 CM1'的编码数据流 ST_{NEW} 从数据流服务器 49 中复制，从而省略了大量时间重复对相同商业广告进行编码的过程。在这种情况下，接合过程将按下文中的详细描述来执行。

首先，将描述在地方电台 40 的初始化过程，该过程用于对地方商业广告 CM1'进行编码，以便替代传输节目的商业广告 CM1，以及将编码数据流 ST_{NEW} 存储到数据流服务器 49 中。广播系统控制器 41 控制 CM 服务器 47，以再现商业广告 CM1'的视频数据，以替代传输节目的商业广告 CM1。然后，编码器 481 接收从 CM 服务器 47 复制的基带视频数据，并且将该视频数据的每个画面的编码难度 Di 提供给编码器控制器 480。编码器控制器 480，类似于图 8 中描述的编码器控制器 350，给编码器 481 提供目标比特率，以使编码器 481 生成适当的编码位。编码器 481 基于编码器控制器 480 提供的目标比特率 Ri 执行编码过程，然后生成最佳比特率的编码基本数据流 ST_{NEW}。从编码器 481 输出的编码基本数据流 ST_{NEW} 提供给数据流服务器 49。数据流服务器 49 将数据流状态下的编码基本数据流记录到随机存取记录媒体。这样就完成了将编码数据流 ST_{NEW} 存储到数据流服务器 49 的初始化过程。

然后，将描述接合处理过程，该过程用于接合从关键电台发送的传输节目的编码数据流 ST_{OLD} 和通过以上描述的初始化过程存储在数据流服务器

49 中的编码数据流 ST_{NEW}。

从关键电台 30 发送的编码数据流 ST_{OLD} 由数据流转换电路 44 从传输数据流形式转换成基本数据流形式。转换成基本数据流形式的编码数据流 ST_{OLD} 提供给数据流接合器 50。

如图 20 所示，数据流接合器 50 包括接合控制器 500、切换电路 501、数据流分析电路 502、数据流处理器 503 以及接合电路 504。

在接合过程的第一实施例中，接合控制器 500 将切换电路 501 的输入端切换到“a”，以便将数据流服务器 49 提供的基本数据流 ST_{NEW} 提供给数据流分析电路 502。

数据流分析电路 502 是用于对编码数据流 ST_{OLD} 和编码数据流 ST_{NEW} 的语法进行语法分析和解析的电路。具体讲，数据流分析电路 502 搜索编码数据流 ST_{OLD} 中描述的 32 比特的 picture_start_code，以便了解在数据流中描述有关画面标题的信息的位置，如从图 10 和 15 所示的编码数据流的语法中所理解的。然后，数据流分析电路 502 查找 3 比特的 picture_coding_type，它开始于 picture_start_code 后的 11 比特，以便了解画面类型。此外，数据流分析电路 502 从紧接着 3 比特的 picture_coding_type 所描述的 16 比特的 vbv_delay 中了解编码画面的 vbv_delay。

此外，数据流分析电路 502 搜索在编码数据流 ST_{OLD} 和编码数据流 ST_{NEW} 中描述的 32 比特的 extension_start_code，以便了解在数据流中描述有关画面编码扩展的信息的位置，如从图 10 和 15 所示的编码数据流的语法中所理解的。然后，数据流分析电路 502 搜索 1 比特的 top_field_first 和 repeat_first_field，以便了解编码画面的帧结构，所述 top_field_first 描述在 picture_start_first 之后的 25 比特，而所述 repeat_first_field 描述在 top_field_first 之后的 6 比特。例如，如果编码画面的“top_field_first”为“1”，它表示顶场暂时超前底场的帧结构。如果“top_field_first”为“0”，它表示顶场暂时落后底场的帧结构。如果编码数据流中的标志“top_field_first”为“1”并且“repeat_first_field”为“1”，它表示可因此在解码时从顶场生成重复场的画面结构。如果编码数据流中的标志“top_field_first”为“0”并且“repeat_first_field”为“1”，它表示可因此在解码时从底场生成重复场的画面结构。

上述的 picture_coding_type、vbv_delay、top_field_first、以及 repeat_first_field 都是从用于每个画面的编码数据流提取的，并且提供给接合

控制器 500。提供给数据流分析电路 502 的基本数据流 ST_{OLD} 和基本数据流 ST_{NEW} 直接作为基本数据流 ST_{OLD} 和基本数据流 ST_{NEW} 提供给数据流处理器 503。

另外，数据流分析电路 502 具有用于计算所提供的数据流 ST_{OLD} 和数据流 ST_{NEW} 的比特数的计数器。基于计数值和每个画面的生成比特量，数据流分析电路 502 模拟分析关于每个画面的 VBV 缓冲器中的数据剩余量。由数据流分析电路 502 计算的关于每个画面的 VBV 缓冲器中的数据剩余量也提供给接合控制器 500。

数据流处理器 503 是用于改变数据流结构、数据流 ST_{OLD} 和数据流 ST_{NEW} 的数据单元和标志的电路，以使由数据流 ST_{OLD} 和数据流 ST_{SPL} 接合生成的接合数据流 ST_{SPL} 变为无缝数据流。现在将参照图 21A 到 21B 描述由数据流处理器 503 执行的特定处理过程。

图 21A 显示了从关键电台 30 提供的原始数据流 ST_{OLD}，和 VBV 缓冲器中的数据流 ST_{OLD} 的数据占有量的轨迹。图 21B 显示了存储在数据流服务器 49 中的替代数据流 ST_{NEW}，和 VBV 缓冲器中的数据流 ST_{NEW} 的数据占有量的轨迹。图 21C 显示了通过接合接合点 SP1 和 SP2 的数据流 ST_{OLD} 和数据流 ST_{NEW} 所得到的接合数据流 ST_{SPL}，以及 VBV 缓冲器中的接合数据流 ST_{SPL} 的数据占有量的轨迹。在图 21A 中，示出了 DTS(解码时间标记)。SP1vbv 表示在 VBV 缓冲器中的数据占有量的轨迹上的第一接合点。SP2vbv 表示在 VBV 缓冲器中的数据占有量的轨迹上的第二接合点。VO(I6)表示当画面 B5 从 VBV 缓冲器中导出时在 VBV 缓冲器上缓冲的画面 I6 的数据量。GB(I6)表示画面 I6 的生成的比特量。VD(I6)表示画面 I6 的 vbv_delay 的值。VO(B7)表示当画面 I6 从 VBV 缓冲器中导出时在 VBV 缓冲器上缓冲的画面 B7 的数据量。GB(B11)表示画面 B11 的生成的比特量。VD(I12)表示画面 I12 的 vbv_delay 的值。VO(I12)表示当 B11 从 VBV 缓冲器中导出时在 VBV 缓冲器中缓冲的画面 I12 的数据量。在图 21B 中，GB(I6')表示画面 I6'的生成的比特量。VD(I6')表示画面 I6'的 vbv_delay 的值。VO(I6')表示在 VBV 缓冲器中的第一接合点 SP1vbv 在 VBV 缓冲器上缓冲的画面 I6'的数据量。GB(B11')表示画面 B11'的生成的比特量。VO(I12')表示在 VBV 缓冲器中的第二接合点 SP2vbv 在 VBV 缓冲器上缓冲的画面 B12'的数据量。在图 21C 中，GB(I6")表示画面 I6"的生成的比特量，对它进行处理可使接合数据流 ST_{SPL} 变为无

缝数据流。VD(I6")表示画面 I6"的 vbv_delay 的值。GB(B11")表示画面 B11"的生成的比特量，对它进行处理可使接合数据流 ST_{SPL} 变为无缝数据流。

原始数据流 ST_{OLD} 是在关键电台 30 编码的数据流，而替代数据流 ST_{NEW} 是在地方电台 40 编码的数据流。因此，数据流 ST_{OLD} 和数据流 ST_{NEW} 是通过它们各自的视频编码器分别编码的。也就是说，数据流 ST_{OLD} 中的第一画面 I6 的 vbv_delay 的值 VD(I6)与数据流 ST_{NEW} 中的第一数据流 I6'的 vbv_delay 的值 VD(I6')并不相同。在这种情况下，在 VBV 缓冲器中的数据流接合点 SP1vbv 的时刻，VBV 缓冲器中的原始数据流 ST_{OLD} 的数据占有量 VO(I6)与 VBV 缓冲器中的替代数据流 ST_{NEW} 的数据占有量 VO(I6')互不相同。

如本发明的背景技术中所描述的，如果数据流 ST_{OLD} 和数据流 ST_{NEW} 在接合点 SP1 简单接合，则 VBV 缓冲器中的简单接合的数据流的数据占有量变得不再连续或生成上溢/下溢。

因此，在数据流接合器 50 中，数据流处理器 503 根据由数据流分析电路 502 从数据流 ST_{OLD} 和数据流 ST_{NEW} 提取的数据单元，对于提供的数据流 ST_{OLD} 和数据流 ST_{NEW} 的数据流结构执行数据流处理，以使接合数据流 ST_{SPL} 在接合点变得无缝。这个过程将在下文中描述。

接合控制器 500 从数据流分析电路 502 接收信息，诸如每个画面的 picture_coding_type、vbv_delay、top_field_first 和 repeat_first_field ，每个画面中生成的比特量以及关于每个画面的 VBV 缓冲器中的数据占有量，并将它们作为关于数据流 ST_{OLD} 的数据单元。在图 21A 到 21C 中，画面 I6 的 vbv_delay 的值表示为 VD(I6)，而画面 I6 生成的比特量表示为 GB(I6)。VBV 缓冲器中的画面 I6 的数据占有量表示为 VO(I6)。

接下来，将描述由接合控制器 500 和数据流处理器 503 在接合点 SP1 关于 VBV 缓冲器执行的处理过程。

首先，如果接合控制器 500 已确定在接合点 SP1 的原始数据流 ST_{OLD} 的画面 I6 的 vbv_delay 的值 VD(I6)和替代数据流 ST_{NEW} 的画面 I6'的 vbv_delay 的值 VD(I6')互不相同，接合控制器 500 指示数据流处理器 503 将替代数据流 ST_{NEW} 中描述的画面 I6'的 vbv_delay 的值从 VD(I6')写入 VD(I6)。

按照来自接合控制器 500 的命令，数据流处理器 503 将替代数据流 ST_{NEW} 的画面标题中描述的 16 比特的 vbv_delay 的值从 VD(I6')重写到

VD(I6)。

在这种情况下，如果替代数据流 ST_{NEW} 中的 vbv_delay 的值简单地从 VD(I6')重写入 VD(I6)，以便按照重写的 vbv_delay 从 VBV 缓冲器导出比特流，则画面 I6'生成的比特量不足，从而造成 VBV 缓冲器的下溢。这样，接合控制器 500 执行将填充字节插入替代数据流 ST_{NEW} 的画面 I6'中的过程，以使替代数据流 ST_{NEW} 的画面 I6'生成的比特量 GB(I6')变为无缝接合数据流 ST_{SPL} 的画面 I6"生成的比特量 GB(I6")。填充字节是由虚比特"0"组成的。

为执行插入填充字节的过程，接合控制器 500 使用作为与数据流 ST_{OLD} 中的画面 I6 和画面 B7 有关的信息接收的 VBV 缓冲器中的生成的比特量 GB(I6)和数据占有量 VO(I6)、作为与数据流 ST_{NEW} 中的画面 I6'有关的信息接收的 VBV 缓冲器中的生成的比特量 GB(I6')和数据占有量 VO(I6')，来计算要插入的填充字节的数据量。具体讲，填充字节 SB1[字节]是基于以下等式(2)来计算的。

$$SB1[\text{比特}] = \{GB(I6'')-GB(I6')\}/8$$
$$=\{GB(I6)-GB(I6')+VO(B7)-VO(B7')\}/8 \quad (2)$$

接合控制器 500 控制数据流处理器 503 将按照公式(2)计算的填充字节 SB1 插入到数据流 ST_{NEW} 中。

按照来自接合控制器 500 的命令，数据流处理器 503 描述插入数据流 ST_{NEW} 的填充字节 SB1。作为填充字节应在数据流中描述的位置，最理想的位置是在编码数据流 ST_{NEW} 中的画面 I6 的画面标题的开始代码之前，但是，也可使用另一个开始代码之前的位置。

以上描述的控制过程是由接合控制器500和数据流处理器503在接合点 SP1 关于 VBV 缓冲器执行的。

接下来，将描述由接合控制器 500 和数据流处理器 503 在接合点 SP2 关于 VBV 缓冲器执行的控制。

如果数据流 ST_{NEW} 和数据流 ST_{OLD} 是在接合点 SP2 简单接合，则数据流 ST_{NEW} 的最后一个画面 I11'生成的比特量 GB(B11')不足，因而与数据流 ST_{NEW} 的第一画面 I12 的 VBV 缓冲器中的数据占有量的轨迹不连续。结果，产生了 VBV 缓冲器的下溢或上溢。

因此，接合控制器 500 执行将填充字节插入数据流 ST_{NEW} 中的过程，因

而数据流ST_{NEW}的最后画面I11'生成的比特量GB(I11')变为图21中画面I11"生成的比特量GB(I11"),从而实现了在VBV缓冲器中的接合点SP2vbv的VBV缓冲器的连续轨迹。

为执行插入填充字节的过程,接合控制器500使用作为与数据流ST_{OLD}中的画面I12有关的信息接收的VO(I12)、数据流ST_{NEW}的最后画面B11'的生成比特量GB(B11')以及关于数据流ST_{NEW}的画面I12'的VBV缓冲器中的数据占有量VO(I12'),来计算要插入的填充字节的数据量。具体讲,填充字节SB2[字节]基于以下等式(3)来计算。

$$SB2[字节] = \{GB(B11") - GB(B11')\}/8$$
$$= \{VO(I12') - VO(I12)\}/8 \quad (3)$$

换句话说,数据占有量VO(I12')是在最后画面B11'从VBV缓冲器中导出之后与数据流ST_{NEW}有关的VBV缓冲器中的数据占有量。数据占有量VO(I12')能容易地由数据流分析电路502检测,数据流分析电路502通过计算数据流ST_{NEW}的比特数来掌握VBV的轨迹。

接合控制器500控制数据流处理器503将按照公式(3)计算的填充字节SB2插入到数据流ST_{NEW}中。

按照来自接合控制器500的命令,数据流处理器503将填充字节SB2描述为关于数据流ST_{NEW}的画面B11'的信息。作为填充字节应被描述在数据流中的位置,最理想的位置是在编码数据流ST_{NEW}中的画面B11'的画面标题的开始代码之前。

以上描述的处理过程的控制是由接合控制器500和数据流处理器503在接合点SP2相对于VBV缓冲器执行的。

参照图22A到22C,现在将描述由接合控制器500和数据流处理器503在接合点SP1相对于诸如top_field_first和repeat_first_field等的标志执行的处理的第一个例子。

图22A显示了由在关键电台30制作的节目1、商业广告CM1和节目2所组成的电视节目PG_{OLD}的帧结构,以及通过对该电视节目PG_{OLD}编码得到的编码数据流ST_{OLD}。图22B显示了在地方电台40制作的替代商业广告CM1'的帧结构,以及通过对该替代商业广告CM1'编码而得到的编码数据流ST_{NEW}。图22C显示了通过接合原始数据流ST_{OLD}和替代数据流ST_{NEW}而生

成的接合数据流 ST_{SPL}，以及对该接合数据流 ST_{SPL} 解码而得到的帧结构。

接合控制器 500 将数据流分析电路 502 提供的数据流 ST_{OLD} 中的商业广告 CM1 的每个画面的 top_field_first 与替代数据流 ST_{NEW} 中的商业广告 CM1' 的 top_field_first 作比较。如果数据流 ST_{OLD} 的 top_field_first 与替代数据流 ST_{NEW} 的 top_field_first 相同，则使用相同的字段结构，因此不再需要对于诸如 top_field_first 和 repeat_first_field 的标志的处理过程。然而，如图 22A 到 22C 所示，如果原始商业广告 CM1 的 top_field_first 为"0"而替代商业广告 CM1'的 top_field_first 为"1",则将发生图 6 中所描述的场的中断和复制问题。

因此，本发明的数据流接合器 50 对接近接合点的画面的 top_field_first 和 repeat_first_field 进行重写，以使不产生由接合过程生成的场的漏失和重叠的不符合 MPEG 标准的数据流。

在图 22A 到 22C 所示的例子中,接合控制器 500 控制数据流处理器 503 从 0 到 1 重写画面 P3 的 repeat_first_field,画面 P3 具有由顶场 T4 和底场 B4 组成的帧。另外，接合控制器 500 控制数据流处理器 503 从 0 到 1 重写画面 P9'的 repeat_first_field，画面 P9'具有由顶场 t10 和底场 b11 组成的帧，从而在接合点 SP2 生成无缝数据流。此外，因为商业广告 CM1'通过重写画面 P9' 的 repeat_first_field 而从原始商业广告 CM1 移位一个帧的时间，所以接合控制器 500 控制数据流处理器 503 从数据流 ST_{OLD} 删除节目 2 的所有画面中首先显示在显示器上的画面 B13。

按照来自接合控制器 500 的指令,数据流处理器 503 在原始数据流 ST_{OLD} 中搜索关于画面 P3 的 picture_coding_extension 的开始代码，然后从 0 到 1 重写其中的 repeat_first_field 的值。这样，因为重复场 B4'是通过对具有重写的 repeat_first_field 值的画面 P3 进行解码而生成的，因而场在接合点 SP1 变得连续。同样，数据流处理器 503 在替代数据流 ST_{OLD} 中搜索关于画面 P9' 的 picture_coding_extension 的开始代码，并且从 0 到 1 重写其中的 repeat_first_field 的值。这样，因为重复场 t10'是通过对具有重写的 repeat_first_field 值的画面 P9'进行解码而生成的，因而场在接合点 SP2 变得连续。同样，数据流处理器 503 删除原始数据流 ST_{OLD} 中的一部分，在该部分描述与画面 B13 有关的数据单元，或者用空数据替代该部分。

图 23A 到 23C 显示了图 22A 到 22C 中所描述的对于诸如 top_field_first 和 repeat_first_field 等标志进行处理的另一个示例。参照图 23A 到 23C，将

描述由接合控制器 500 和数据流处理器 503 在接合点 SP1 和 SP2 对诸如 top_field_first 和 repeat_first_field 等标志进行处理的第二个例子。

在图 23A 到 23C 所示的例子中，接合控制器 500 控制数据流处理器 503 从 1 到 0 重写由顶场 t5 和底场 b6 组成的画面 B7'的 top_field_first，并且从 0 到 1 重写画面 B7'的 repeat_first_field，以使在接合点 SP1 的节目 1 和商业广告 CM1'之间的衔接口的场变为连续。另外，接合控制器 500 控制数据流处理器 503 从 1 到 0 重写由顶场 T11 和底场 B11 组成的画面 B13 的 top_field_first，以使接合点 SP2 的商业广告 CM1'和节目 2 之间的衔接口的场变得连续。而且，接合控制器 500 控制数据流处理器 503 从 1 到 0 重写由顶场 T12 和底场 B12 所组成的画面 B14 的 top_field_first，并且从 1 到 0 重写 repeat_first_field。

在接合控制器 500 的控制下，数据流处理器 503 在替代数据流 ST_{NEW} 中搜索关于画面 B7'的 picture_coding_extension 的开始代码。然后数据流处理器 503 从 1 到 0 重写数据流中的 top_field_first 并且从 0 到 1 重写 repeat_first_field。这样，通过对具有重写的 top_field_field 和 repeat_first_field 的值的画面 B7'进行解码，底场 b6 的显示时间移位了一个帧，并且生成重复场 b6'。因而场在接合点 SP1 变为连续。同样，数据流处理器 503 在原始数据流 ST_{OLD} 中搜索关于画面 B13 的 picture_coding_extension 的开始代码，并且从 1 到 0 重写其中的 top_field_first。另外，数据流处理器 503 从 1 到 0 重写关于原始数据流 ST_{OLD} 中的画面 B14 的 top_field_first，并且从 1 到 0 重写 repeat_first_field。这样，因为通过对具有重写的 top_field_first 和 repeat_first_field 的值的画面 B13 和 B14 进行解码，使底场 B11 和 B12 的显示时间移位一个帧，所以场在接合点 SP2 变得连续。

比较一下图 22A 到 22C 所示的处理的第一个例子和图 23A 到 23C 所示的处理的第二个例子。从图 22C 可知，因为在替代商业广告 CM1'的开端显示的画面 B7 从在原始商业广告 CM1 的开端显示的画面 B7 移位了一个场，所以替代商业广告 CM1'的显示时间延迟了一个场。一个场的显示延迟很难用肉眼看出。然而，因为广播电台通过广播客户公司的商业广告而获得利润，所以可能会要求广播电台在没有任何延迟的情况下精确地广播商业广告，而不是广播诸如节目 1 的主要节目。在这种精确要求显示时间的情况下，图 23A 到 23C 所示的处理的第二个例子是有效的。如同图 23A 到 23C 所示的处理

的第二个例子，通过重写画面 B7'的 top_field_first 和 repeat_first_field 的值，替代商业广告 CM1'的第一画面 B7'能精确地显示，而没有任何从原始商业广告 CM1 的第一个显示画面的延迟。

总之，从数据流处理器 503 输出的数据流 ST_{NEW} 的 VBV 缓冲器中的数据占有量的轨迹与 VBV 缓冲器中的数据流 ST_{OLD} 的数据流占有量的轨迹是一致的，并且具有相对于场图形/帧图形的一致性。因此，通过基于来自接合控制器 500 的控制信号对接合电路 504 的切换操作进行控制，通过连接接合点 SP1 的数据流 ST_{OLD} 之后的数据流 ST_{NEW} 和连接接合点 SP2 的数据流 ST_{NEW} 之后的数据流 ST_{OLD} 生成了接合数据流 ST_{SPL}。在接合数据流 ST_{SPL} 中的接合点 SP1 和 SP2，VBV 缓冲器中的数据占有量的轨迹是连续的并且场图形/帧图形也是连续的。

现在将描述接合过程的第二个实施例。接合过程的第二个实施例是关于在下列情况下执行的接合过程：当传输节目的编码数据流 ST_{OLD} 从关键电台 30 发送时，要被最新插入的商业广告 CM1'被编码生成编码数据流 ST_{NEW}。即，分析从关键电台 30 发送的传输节目的编码数据流 ST_{OLD}，并且基于分析结果对最新插入的商业广告 CM1'进行编码。

首先，从关键电台 30 发送的编码数据流 ST_{OLD} 由数据流转换电路 44 从传输数据流形式转换成基本数据流换形式。转换成基本数据流形式的编码数据流 ST_{OLD} 提供给数据流接合器 50 的数据流分析电路 502。

数据流接合器 50 的数据流分析电路 502 是用于分析编码数据流 ST_{OLD} 的数据流语法的电路。在接合过程的第二个实施例中，数据流分析电路 502 仅分析编码数据流 ST_{OLD} 的语法而不分析替代数据流 ST_{NEW} 的语法。

具体讲，数据流分析电路 502 搜索原始数据流 ST_{OLD} 中描述的 32 比特的 picture_start_code，以便了解与画面标题有关的信息在数据流中描述的位置，如从图 10 和 15 中所示的编码数据流的语法中所理解的。然后，数据流分析电路 502 搜索开始于 picture_start_code 之后的 11 比特的 3 比特的 picture_coding_type，以便了解画面类型。而且，数据流分析电路 502 从紧接着 3 比特的 picture_coding_type 所描述的 16 比特的 vbv_delay 中，了解编码画面的 vbv_delay。

另外，数据流分析电路 502 搜索编码数据流 ST_{OLD} 中描述的 32 比特的 extension_start_code，以便了解数据流中与画面编码扩展有关的信息的描述

位置，如从图 10 和 15 中所示的编码数据流的语法中所理解的。然后，数据流分析电路 502 搜索在 picture_start_code 之后 25 比特描述的 1 比特的 top_field_first，和在 top_field_first 之后 6 比特描述的 repeat_first_field，以便了解编码画面的帧结构。

数据流分析电路 502 通过接合控制器 500 给广播系统控制器 41 提供数据单元，诸如从原始数据流 ST_{OLD} 为每个画面提取的 picture_coding_type、vbv_delay、top_field_first 以及 repeat_first_field。没有必要发送原始数据流 ST_{OLD} 的所有画面的数据单元。仅发送诸如对应于发送节目的商业广告 CM1 的画面的 picture_coding_type、vbv_delay、top_field_first 以及 repeat_first_field 就足够了。

广播系统控制器 41 控制 CM 服务器 47，以再现 CM1'的视频数据，从而取代传输节目的商业广告 CM。另外，广播系统控制器 41 给编码器块 48 的编码控制器 480 提供从原始数据流 ST_{OLD} 提取的 picture_coding_type、vbv_delay、top_field_first 以及 repeat_first_field。

编码器控制器 480 控制编码器 481 利用从广播系统控制器 41 提供的 picture_coding_type、vbv_delay、top_field_first 和 repeat_first_field 对替代商业广告 CM1'的基带视频数据进行编码。也就是说，对替代商业广告 CM1'进行编码，以使原始商业广告 CM1 的编码数据流 ST_{OLD} 的 picture_coding_type、vbv_delay、top_field_first 和 repeat_first_field 与通过对替代商业广告 CM1'进行编码得到的数据流 ST_{NEW} 的 picture_coding_type、vbv_delay、top_field_first 和 repeat_first_field 变得彼此完全相同。结果，生成了具有与原始商业广告 CM1 的编码数据流 ST_{OLD} 的 picture_coding_type、vbv_delay、top_field_first 和 repeat_first_field 完全相同的 picture_coding_type、vbv_delay、top_field_first 和 repeat_first_field 的编码数据流 ST_{NEW}。

接合控制器 500 将切换电路 501 的输入端切换到"b"，以便将从编码器 481 输出的基本数据流 ST_{NEW} 提供给数据流分析电路 502。在接合过程的第二个实施例中，因为数据流分析电路 502 仅分析编码数据流 ST_{OLD} 的语法而不分析替代数据流 ST_{NEW} 的语法，数据流 ST_{NEW} 在没有由数据流分析电路 502 进行分析的情况下原样输出。

在接合过程的第二个实施例中，因为不需要用于改变从数据流分析电路 502 输出的数据流 ST_{OLD} 和数据流 ST_{NEW} 中的数据单元的数据流处理，数据

流处理器503仅执行用于实现数据流ST_{OLD}和数据流ST_{NEW}之间的帧同步的同步过程(或帧同步)。具体讲，数据流处理器503有用于若干帧的FIFO缓冲器，并且将数据流ST_{OLD}缓冲到FIFO缓冲器中，直到替代数据流ST_{NEW}从编码器481输出。因而能实现数据流ST_{OLD}和数据流ST_{NEW}之间的帧同步。被处理用于帧同步的数据流ST_{OLD}和数据流ST_{NEW}提供给接合电路504。

接合控制器500控制接合电路504的切换，以便使数据流ST_{NEW}在接合点SP1连接到数据流ST_{OLD}之后，并且使数据流ST_{OLD}在接合点SP2连接到数据流ST_{NEW}之后。结果，从接合电路504输出接合数据流ST_{SPL}。

尽管数据流ST_{OLD}和数据流ST_{NEW}是由接合电路504简单切换的，VBV缓冲器中的接合数据流的数据占有量的轨迹是连续的而且接合点的帧图形也是连续的。其原因如下。即，因为数据流ST_{NEW}是基于对原始数据流ST_{OLD}的语法分析结果被编码的，所以生成了同原始数据流ST_{OLD}一致的数据流ST_{NEW}。因此，VBV缓冲器中的接合数据流ST_{SPL}的轨迹与VBV缓冲器中的原始数据流的数据占有量的轨迹完全相同，而且生成的接合数据流ST_{SPL}的帧结构与原始数据流ST_{OLD}的帧结构相同完全。

因此，在第二个实例中，对从关键电台发送的原始编码数据流ST_{OLD}的语法进行分析，并且根据分析结果对替代商业广告CM1'进行编码，使其具有与编码数据流ST_{OLD}相同的数据流结构和编码参数。因此，在接合分别生成的编码数据流ST_{OLD}和编码数据流ST_{NEW}的过程中，能容易地实现编码数据流ST_{OLD}和编码数据流ST_{NEW}之间的一致性。结果，能生成符合MPEG标准的无缝接合的数据流ST_{SPL}。

现在将描述接合过程的第三个实施例。在接合过程的第三个实施例中，用于对原始商业广告CM1的编码数据流和替代商业广告CM1'的编码数据流进行编码的编码参数，是在生成原始商业广告CM1的编码数据流ST_{OLD}和替代商业广告CM1'的编码数据流ST_{NEW}之前提前决定的，并且原始商业广告CM1和替代商业广告CM1'是基于预定的编码参数进行编码的。例如，编码参数是由已经描述的picture_coding_type、vbv_delay、top_field_first、repeat_first_field和生成的比特量所表示的信息。

首先，就用于对原始商业广告CM1进行编码的编码参数来说，picture_coding_type、vbv_delay、top_field_first和repeat_first_field是在关键电台30决定的。关键电台30的广播系统控制器31将编码参数提供给MPEG

编码器块 35 的编码器控制器 350，也将编码参数通过通信线路提供给每个地方电台 40 的广播系统控制器 41。

编码控制器 350 控制视频编码器 351-1V 通过使用从广播系统控制器 31 提供的诸如 picture_coding_type、vbv_delay、top_field_first 和 repeat_first_field 等的编码参数对原始商业广告 CM1 的视频数据进行编码。也就是说，从视频编码器 351-1V 输出的编码数据流 ST_{OLD} 是基于诸如 picture_coding_type、vbv_delay、top_field_first 和 repeat_first_field 等的编码参数的数据流。

从视频编码器 351-1V 输出的编码数据流 ST_{OLD} 通过多路复用器 36 和调制电路 37 提供给地方电台 40。

另一方面，地方电台 40 将诸如从关键电台 30 的广播系统控制器 31 提供的 picture_coding_type、vbv_delay、top_field_first 和 repeat_first_field 等的编码参数提供给编码器块 48 的编码器控制器 480。

编码控制器 480 控制编码器 481 使用从广播系统控制器 41 提供的 picture_coding_type、vbv_delay、top_field_first 和 repeat_first_field 等的编码参数对替代商业广告 CM'的基带视频数据进行编码。也就是说，生成具有与原始商业广告 CM 的编码数据流 ST_{OLD} 的 picture_coding_type、vbv_delay、top_field_first 和 repeat_first_field 完全相同的 picture_coding_type、vbv_delay、top_field_first 和 repeat_first_field 的数据流 ST_{NEW}。

接合控制器 500 将切换电路 501 的输入端切换到“b”，以便将从编码器 481 输出的基本数据流 ST_{NEW} 提供给数据流分析电路 502 中。在接合过程的第三个实施例中，数据流分析电路 502 并不执行对编码数据流 ST_{OLD} 和编码数据流 ST_{NEW} 的语法分析。

在接合过程的第三个实施例中，因为不需要用于改变从数据流分析电路 502 输出的数据流 ST_{OLD} 和数据流 ST_{NEW} 中的数据单元的数据流处理，所以数据流处理器 503 仅执行用于实现数据流 ST_{OLD} 和数据流 ST_{NEW} 之间的帧同步的同步过程(或帧同步)。具体讲，数据流处理器 503 具有用于多个帧的 FIFO 缓冲器，并且缓冲 FIFO 缓冲器中的数据流 ST_{OLD} 直到替代数据流 ST_{NEW} 从编码器 481 中输出。因此，能实现数据流 ST_{OLD} 和数据流 ST_{NEW} 之间的帧同步。被处理用于帧同步的数据流 ST_{OLD} 和数据流 ST_{NEW} 提供给接合电路 504。

接合控制器 500 控制接合电路 504 的切换，以便使数据流 ST_{NEW} 连接在接合点 SP1 的数据流 ST_{OLD} 之后，并且数据流 ST_{OLD} 连接在接合点 SP2 的数

据流 ST_{NEW} 之后。因此，从接合电路 504 输出接合数据流 ST_{SPL}。

尽管数据流 ST_{OLD} 和数据流 ST_{NEW} 是由接合电路 504 简单切换的，VBV 缓冲器中的接合数据流 ST_{SPL} 的数据占有量的轨迹是连续的且接合点的帧图形也是连续的。这是因为原始商业广告 CM1 和替代商业广告 CM1'是通过使用诸如 picture_coding_type、vbv_delay、top_field_first 和 repeat_first_field 等的编码参数来编码的，这些编码参数是由关键电台 30 的广播系统控制器 31 提前决定的。

因此，在第三个实施例中，编码参数是在关键电台和地方电台之间预定的。然后，在关键电台基于预定的编码参数生成原始商业广告 CM1 的编码数据流 ST_{OLD}，并在关键电台基于预定的编码参数生成替代商业广告 CM1'的编码数据流 ST_{NEW}。因此，在接合分别生成的编码数据流 ST_{OLD} 和编码数据流 ST_{NEW} 的过程中，能容易地实现编码数据流 ST_{OLD} 和编码数据流 ST_{NEW} 之间的一致性。因而能生成符合 MPEG 标准的无缝接合的数据流 ST_{SPL}。

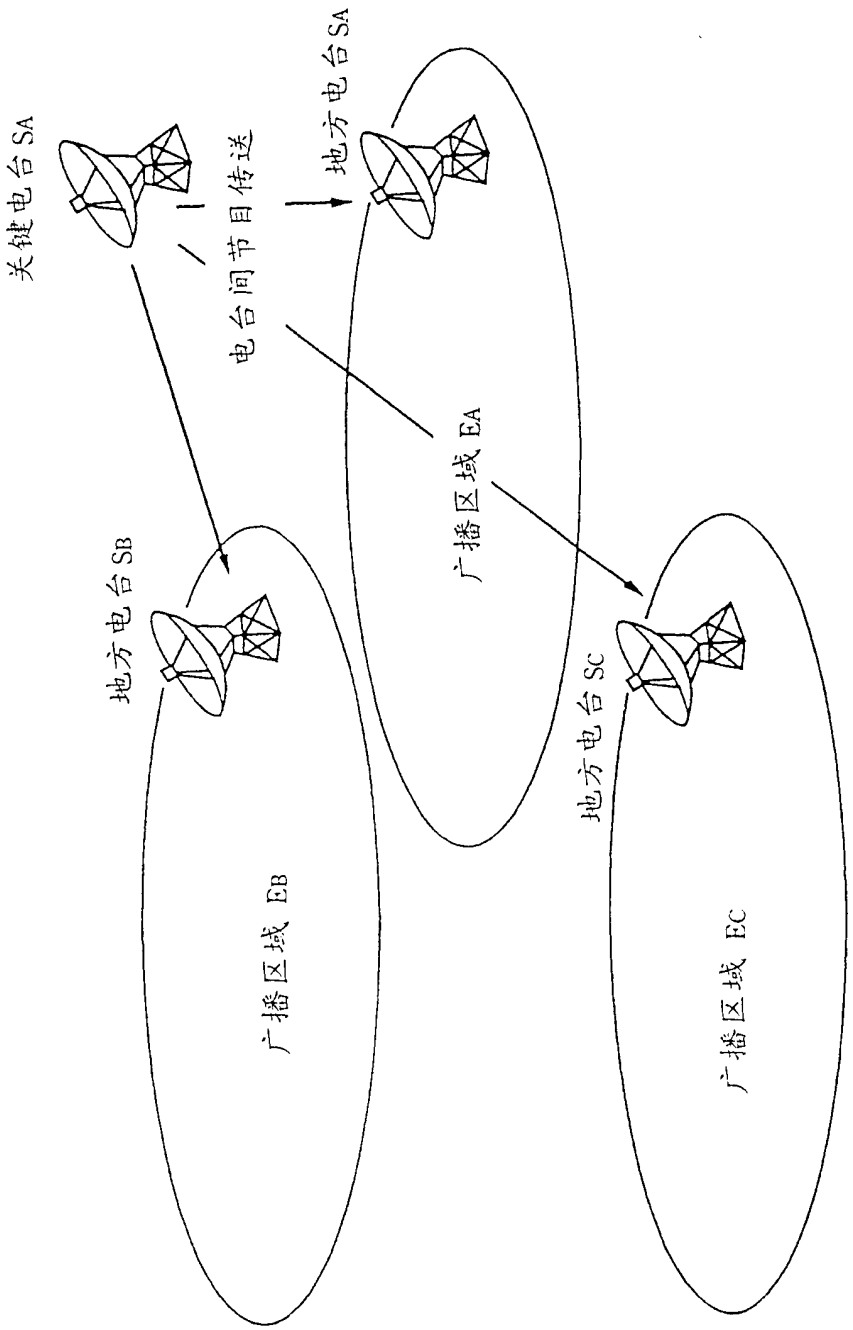


图 1

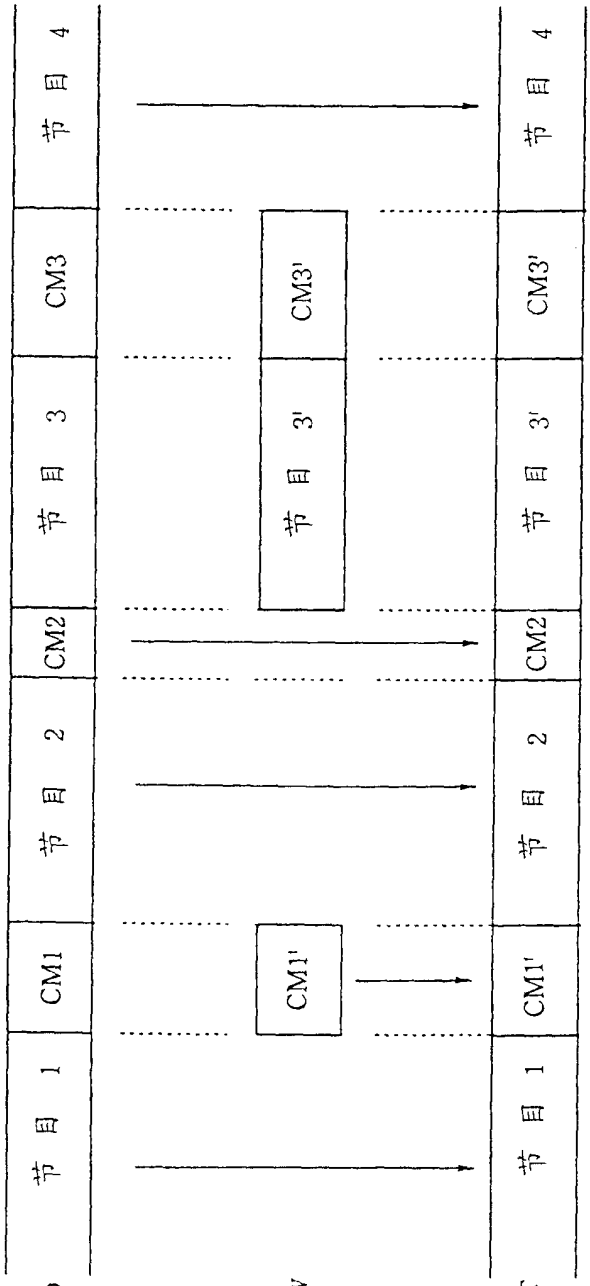

PGOLD
节目 1
CM1
节目 2
CM2
节目 3
CM3
节目 4
PGNEW
CM1'
节目 3'
CM3'
PGEDIT
节目 1
CM1'
节目 2
CM2
节目 3'
CM3'
节目 4

图 2A 图 2B 图 2C

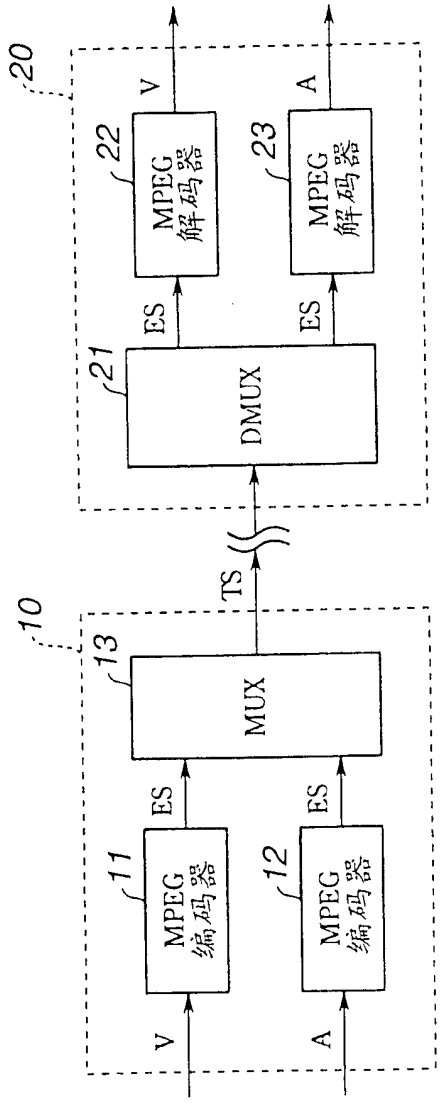

20
V
A
22
MPEG
解码器
23
MPEG
解码器
ES
ES
21
DMUX
TS
10
13
MUX
ES
ES
11
MPEG
编码器
12
MPEG
编码器
V
A

图 3

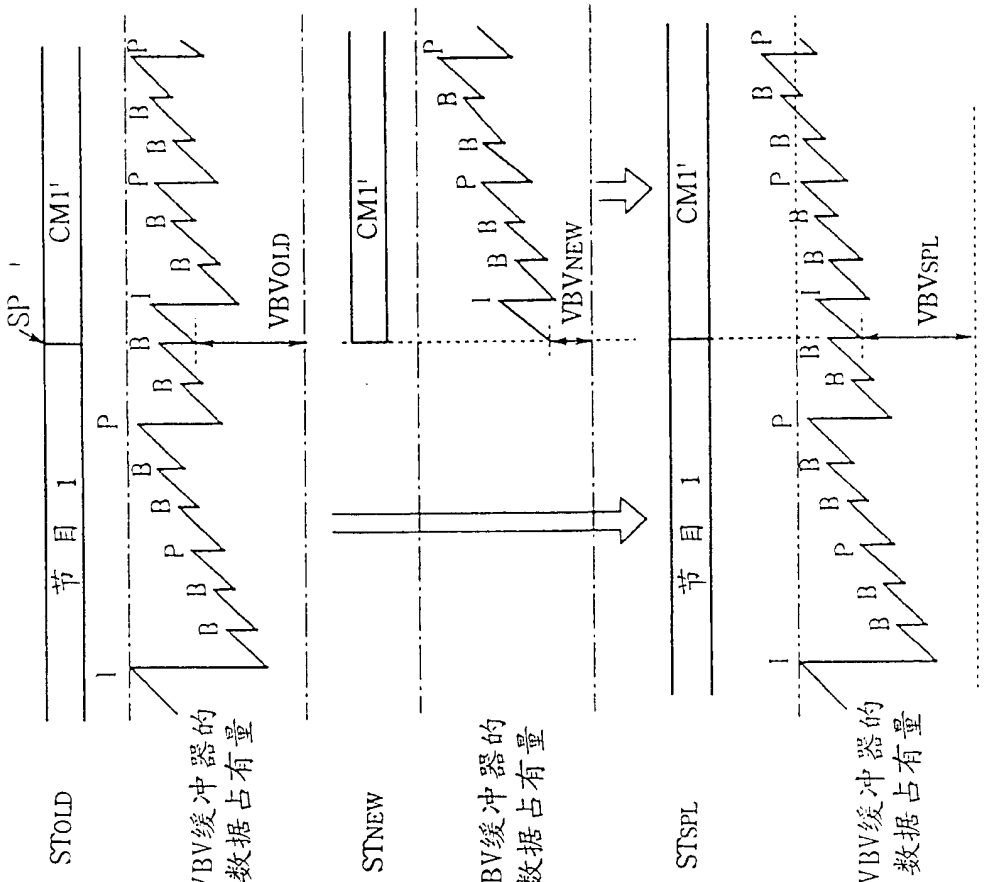

STOLD
节目1
CM1'
SP'
VBV缓冲器的数据占有量
VBVOLD
STNEW
VBVNEW
STSPL
VBVSPL

图 4A

图 4B

图 4C

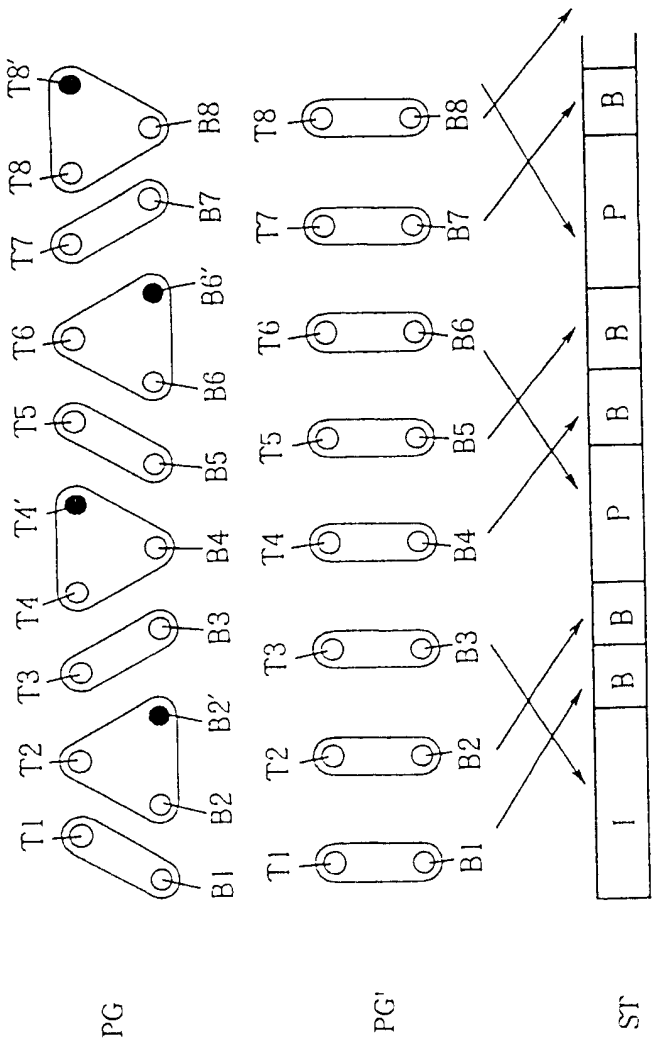

T1
T2
T3
T4
T4'
T5
T6
T7
T8
T8'
B1
B2
B2'
B3
B4
B5
B6
B6'
B7
B8
PG
T1
T2
T3
T4
T5
T6
T7
T8
B1
B2
B3
B4
B5
B6
B7
B8
PG'
I
B
B
P
B
B
P
B
ST

图 5

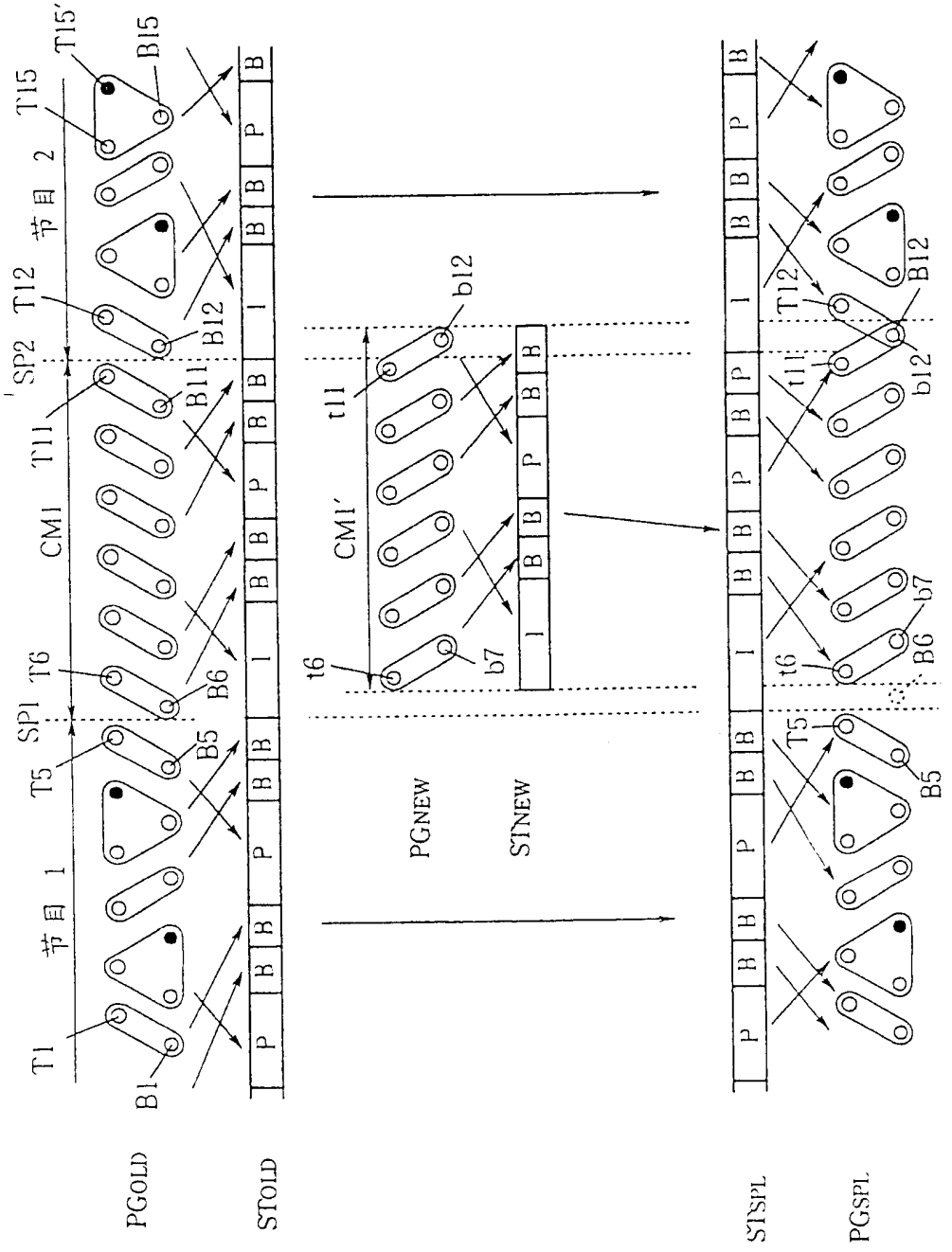

PGOLD
STOLD
PGNEW
STNEW
STSPL
PGSPL
节目 1
节目 2
T1
T5
T6
T11
T12
T15
T15'
SP1
SP2
CM1
CM1'
B1
B5
B6
B11
B12
B15
t6
t11
b7
b12
P
B
I

图 6A 图 6B 图 6C

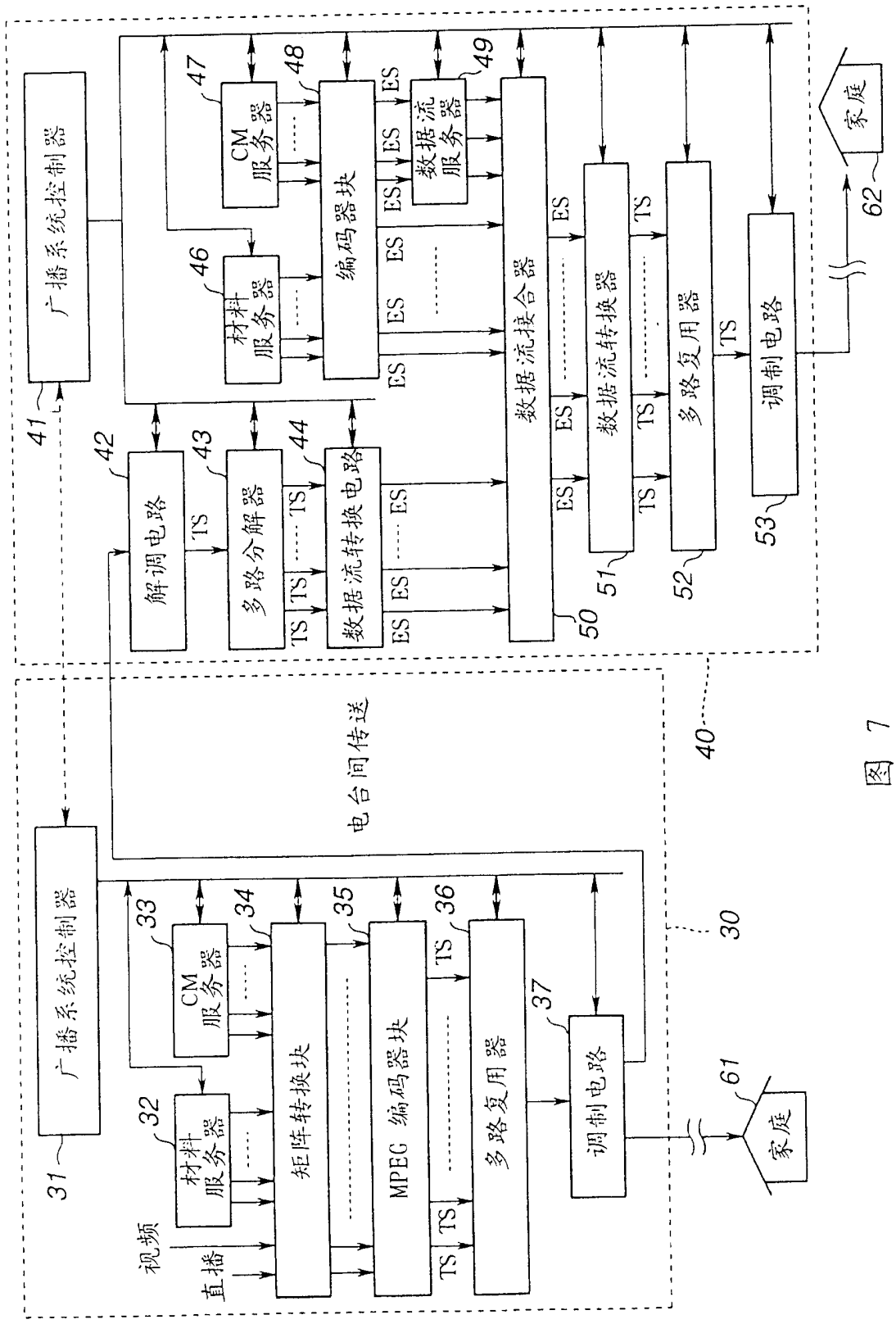

广播系统控制器
CM服务器
编码器块
数据流服务器
材料服务器
数据流接合器
数据流转换器
多路复用器
调制电路
家庭
解调电路
多路分解器
数据流转换电路
电台间传送
矩阵转换块
MPEG 编码器块
视频
直播
TS
ES
30
31
32
33
34
35
36
37
40
41
42
43
44
46
47
48
49
50
51
52
53
61
62

图 7

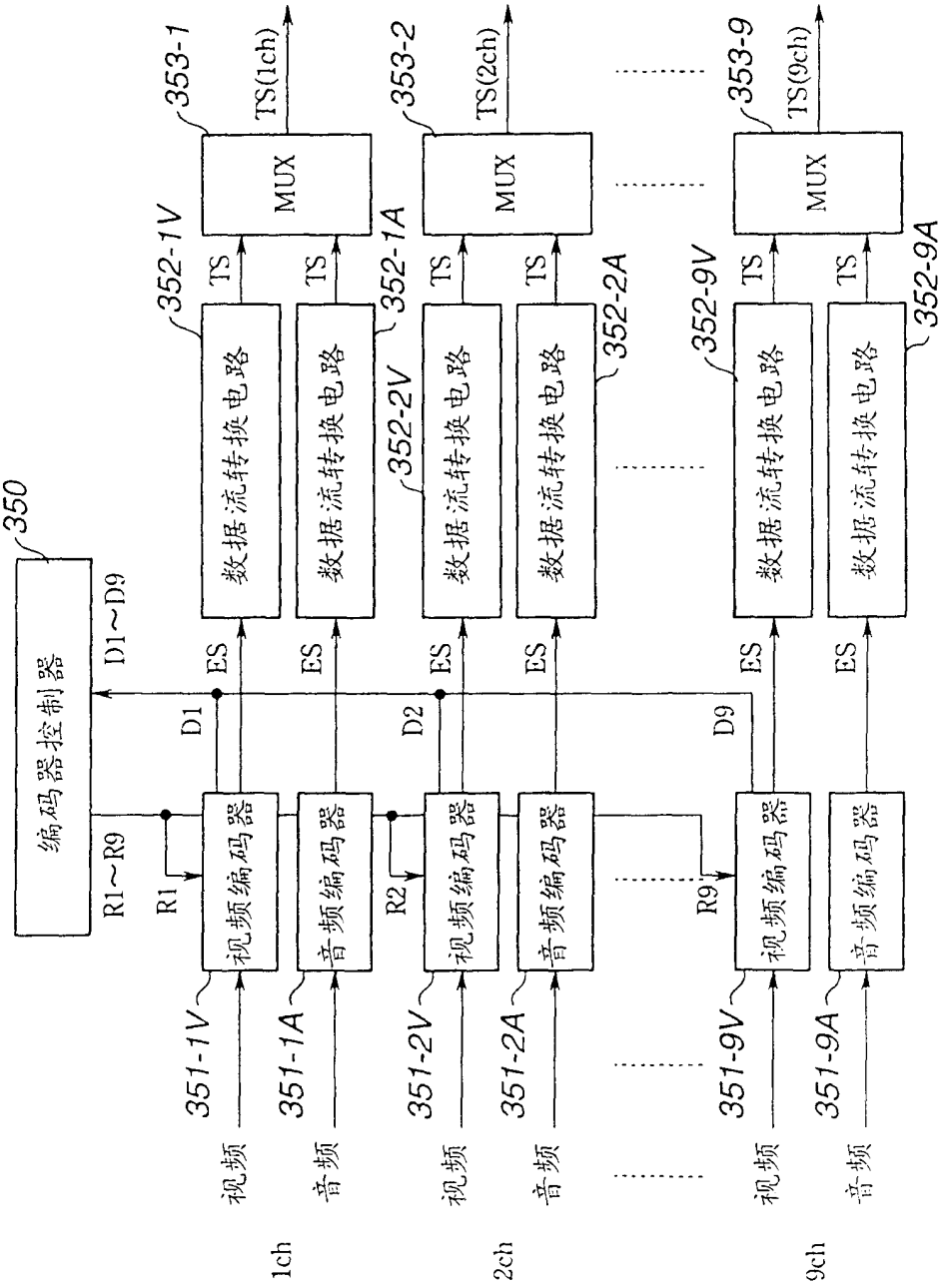

350
编码器控制器
R1～R9
D1～D9
R1
R2
R9
D1
D2
D9
ES
视频编码器
音频编码器
351-1V
351-1A
351-2V
351-2A
351-9V
351-9A
视频
音频
1ch
2ch
9ch
数据流转换电路
352-1V
352-1A
352-2V
352-2A
352-9V
352-9A
TS
MUX
353-1
353-2
353-9
TS(1ch)
TS(2ch)
TS(9ch)

图 8

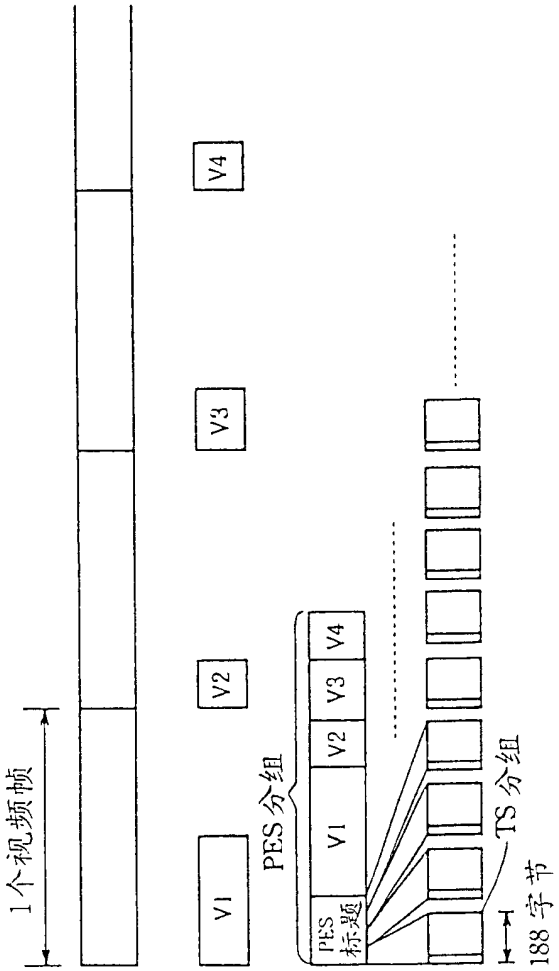


图 9A 图 9B 图 9C 图 9D

video_sequence ( ) {	位数	助记符
next_start_code ( )		
sequence_header ( )		
sequence_extension ( )		
do {		
extension_and_user_data (0)		
do {		
if (nextbits ( ) == group_start_code) {		
group_of_pictures_header ( )		
extension_and_user_data ( )		
}		
picture_header ( )		
picture_coding_extension ( )		
extension_and_user_data (2)		
picture_data ( )		
} while ((nextbits ( ) == picture_start_code)||		
(nextbits ( ) == group_start_code))		
if (nextbits ( ) ! = sequence_end_code) {		
sequence_header ( )		
sequence_extension ( )		
}		
} while (nextbits ( ) ! = sequence_end_code)		
sequence_end_code	32	bslbf
)		

图 10

序列标题

sequence_header( ) {	位数	助记符
sequence_header_code	32	bslbf
horizontal_size_value	12	uimsbf
vertical_size_value	12	uimsbf
aspect_ratio_information	4	uimsbf
frame_rate_code	4	uimsbf
bit_rate_value	18	uimsbf
marker_bit	1	"1"
vbv_buffer_size_value	10	uimsbf
constrained_parameters_flag	1	
load_intra_quantise_matrix	1	
if (load_intra_quantiser_matrix)		
intra_quantiser_matrix[64]	8 * 64	uimsbf
load_non_intra_quantiser_matrix	1	
if (load_non_intra_quantiser_matrix)		
non_intra_quantiser_matrix[64]	8 * 64	uimsbf
next_start_code( )		
}		

图 11

序列扩展

sequence_extension( ) {	位	助记符
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
profile_and_level_indentifier	8	uimsbf
progressive_sequence	1	uimsbf
chroma_format	2	uimsbf
horizontal_size_extension	2	uimsbf
vertical_size_extension	2	uimsbf
bit_rate_extension	12	uimsbf
marker_bit	1	bsalbf
vbv_buffer_size_extension	8	uimsbf
low_dalay	1	uimsbf
frame_rate_extension_n	2	uimsbf
frame_rate_extension_d	5	uimsbf
next_start_code( )		
}		

图 12

扩展数据和用户数据

extension_and_user_data(i) {	位数	助记符
while(((il=1) its( )&&(nextbits( ) extension_start_code)) ||		
(nextbits( )==user_data_start_code)) {		
if(nextbitS( )==extension_start_code)		
extension_data(i)		
if(nextbits( )=user_data_start_code)		
user_data( )		
}		
}		

图 13

画面组标题

group_of_picture_header( ) {	位数	助记符
group_start_code	32	bslbf
time_code	25	bslbf
cloused_gop	1	uimsbf
broken_link	1	uimsbf
next_start_code( )		
}		

图 14

画面标题

picture header( ) {	位数	助记符
picture_start_code	32	bslbf
tenporal_reference	10	uimsbf
picture_coding_type	3	uimsbf
vbv_delay	16	uimsbf
if(picture_coding_type==2||picture_coding__type==3) {		
full_pel_forward_vector	1	
forward_f_code	3	uimsbf
}		
if(picture_coding_type==3) {		
full_pel_backward_vector	1	
backward_f_code	3	uimsbf
}		
while(nextbits( )==' 1') {		
extra_bit_picture／＊with the value " 1" ＊／	1	uimsbf
extra_information_picture	8	
}		
extra_bit_picture／＊with the value " 0" ＊／	1	uimsbf
next_start_code( ) t		
}		

图 15

画面编码扩展

picture_coding_extension( ) {	位数	助记符
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
f_code[0][0]/*forward horizontal*/	4	uimsbf
f_code[0][1]/*forward vertical*/	4	uimsbf
f_code[1][0]/*backward horizontal*/	4	uimsbf
f_code[1][1]/*backward vertical*/	4	uimsbf
intra_dc_precision	2	uimsbf
picture_structure	2	uimsbf
top__field__first	1	uimsbf
frame_pred_frame_dct	1	uimsbf
concealment_motion_vectors	1	uimsbf
q_scale_type	1	uimsbf
intra_vlc_format	1	uimsbf
alternate_scan	1	uimsbf
repeat_first_field	1	uimsbf
chroma_420_type	1	uimsbf
progressive_frame	1	uimsbf
composite_display_flag	1	uimsbf
if(composite_display_flag) {		
v_axis	1	uimsbf
field_sequence	3	uimsbf
sub_carrier	1	uimsbf
burst_amplitude	7	uimsbf
sub_carrier_phase	8	uimsbf
}		
next_start_code( )		
}		

图 16

画面数据

picture_data ( ) {	位数	助记符
do {		
slice ( )		
} while (nextbits ( )==slice_start_code)		
next_start_code ( )		
}		

图 17

语法	位数	助记符
transport packet ( ) {		
sync_byte	8	bslbf
transport_error_indicator	1	bslbf
payload_unit_start_indicator	1	bslbf
transport_priority	1	bslbf
PID	13	uimsbf
transport_scrambling_control	2	bslbf
adaptaion_field_control	2	bslbf
continuity_counter	4	uimsbf
if (adaptation_field_control=='‖ adaptation_field_control==' 11') {		
adaptation field ( )		
}		
if (adaptation_field_control=='‖ adaptation_field_control==' 11') {		
for (i=0 ; 1<N ; i+t) {		
data_byte	8	bslbf
}		
}		
}		

图 18

语法	位数	助记符
adaptation_field( ) {		
adaptation_field_length	8	unimbf
if (adaptation_field_length > 0) {		
discontinuity_indicator	1	bslbf
random access_indicator	1	bslbf
elementary_stream_priority_indicator	1	bslbf
PCR_flag	1	bslbf
OPCR_flag	1	bslbf
splicing_point_flag	1	bslbf
transport_private_data_flag	1	bslbf
adaptation_field_extension_flag	1	bslbf
if (PCR_flag==' 1') {		
program_clock_refernce_base	33	uimsbf
reserved	6	bslbf
progrqam_clock_reference_extension	9	uimsbf
}		
if (OPCR)_flag==' 1') {		
original_program_clock_reference_base	33	uimsbf
reserved	6	bslbf
original_program_clock_reference_extension	9	uimsbf
}		
if (splicing_point_flag==' 1') {		
splice_countdown	8	tcimsbf
}		
if (transport_private_data_flag==' 1') {		
transport_private_data_length	8	uimsbf
for (i=0 ; i< transport_private_data_length ; i+t) {		
provate_data_byte	8	bslbf
}		
}		
if (adaptation_field_extension_flag==' 1') {		
adaptation_field_extension_length	8	unimsbf
ltw_flag	1	bslbf
piecewise_rate_flag	1	bslbf
seamless_splice_flag	1	bslbf
reserved	5	bslbf
if (ltw_flag==' 1') {		
ltw_valid_flag	1	bslbf
ltw_offset	15	unimsbf
}		
if (piecewise_rate_flag==' 1') {		
reserved	2	bslbf
piecewise_rate	22	uimsbf
}		
if (seamless_splice_flag==' 1') {		
splice_type	4	bslbf
DTS_next_AU[32..30]	3	bslbf
marker_bit	1	bslbf
DTS_next_AU[29..15]	15	bslbf
marker_bit	1	bslbf
DTS_next_AU[14..0]	15	bslbf
maker_bit	1	bslbf
}		
for (i=0 ; i<N ; i++) {		
reserved	8	bslbf
}		
for (i=0 ; i<N ; i++) {		
stuffing_byte	8	bslbf
}		
}		
}		

图 19

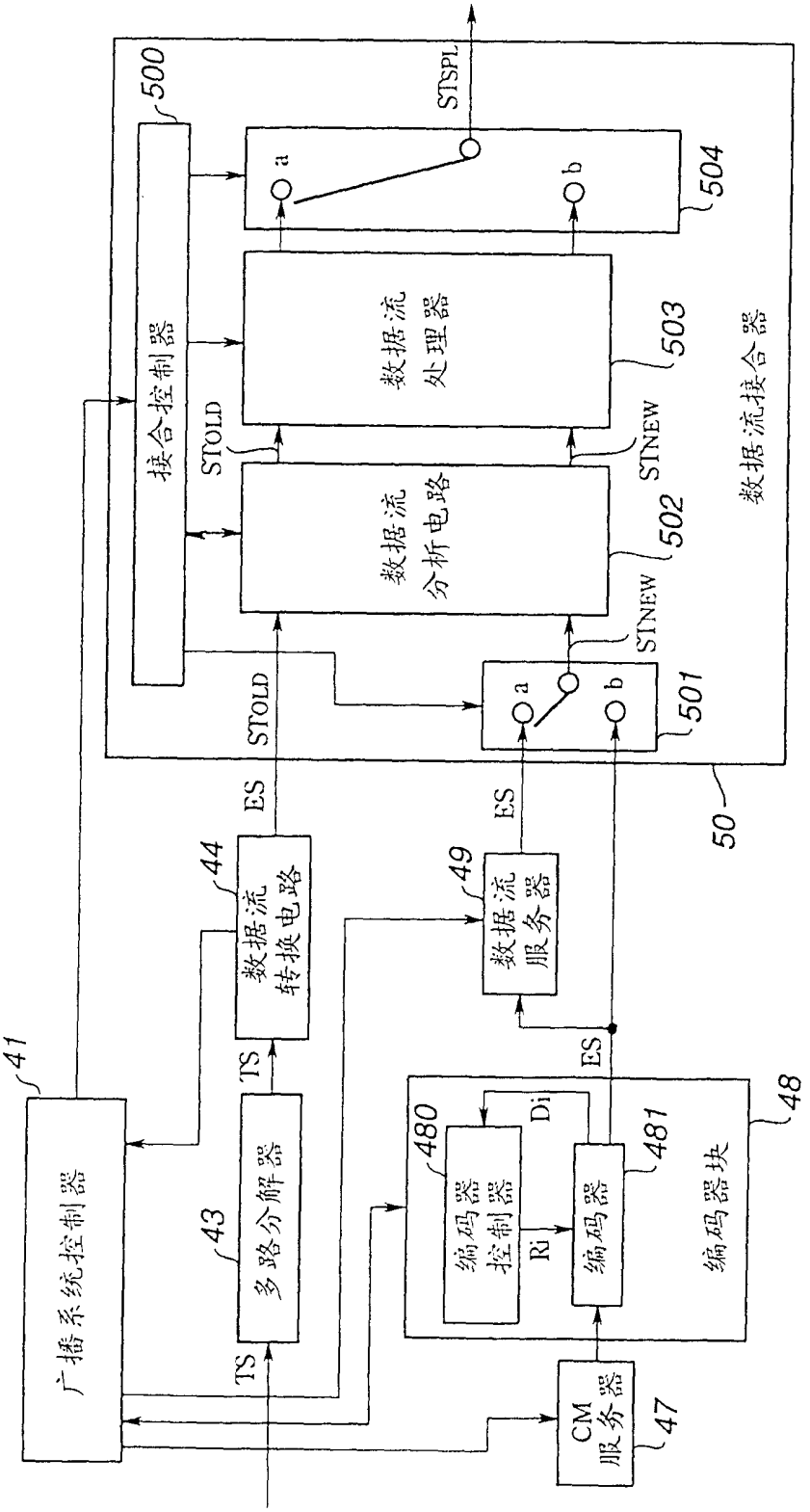

广播系统控制器
41
多路分解器
43
数据流转换电路
44
数据流服务器
49
CM服务器
47
编码器控制器
480
编码器
481
编码器块
48
TS
ES
Di
Ri
接合控制器
500
数据流分析电路
502
数据流处理器
503
a
b
504
501
STOLD
STNEW
STSPL
数据流接合器
50

图 20

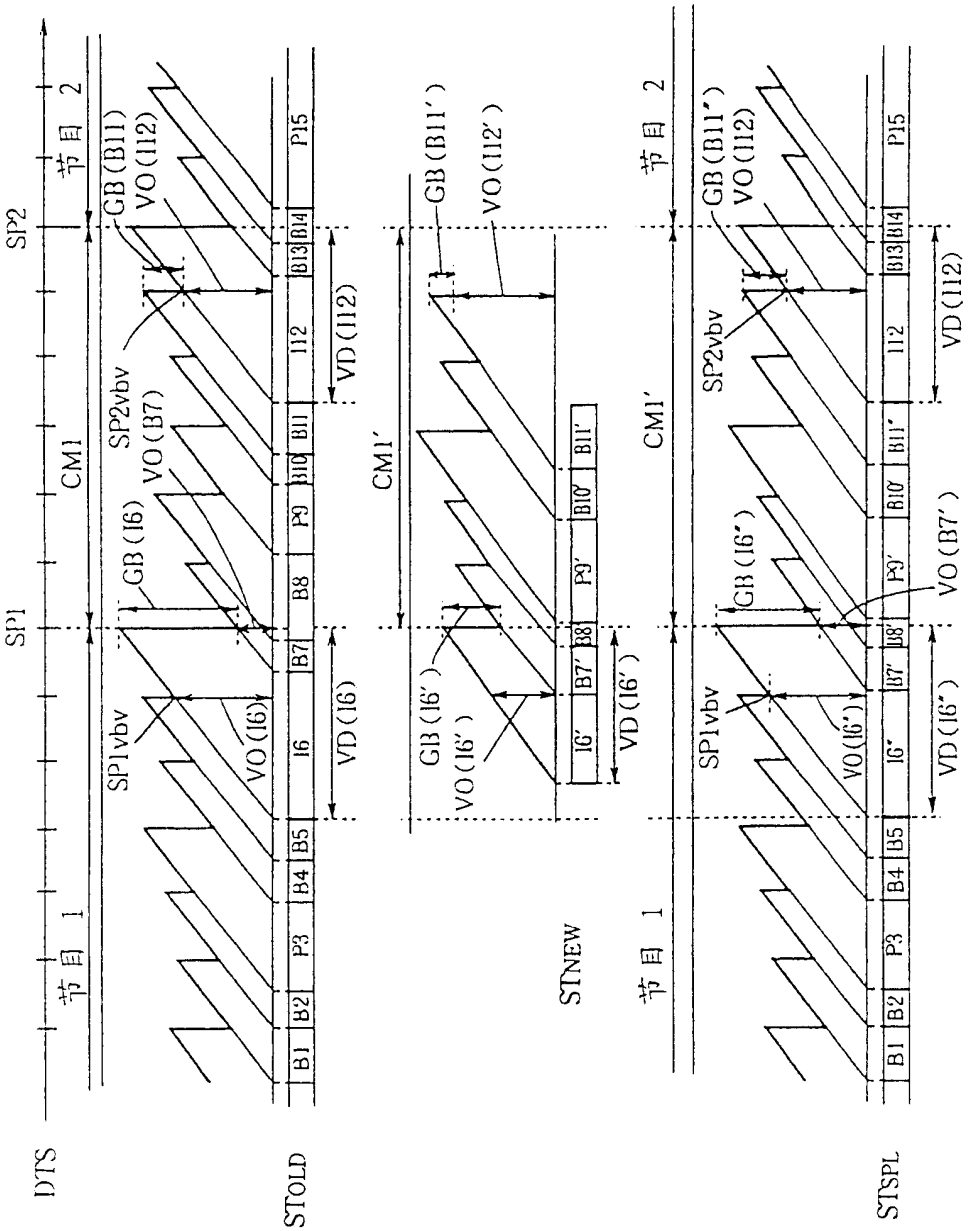


图 21A 图 21B 图 21C

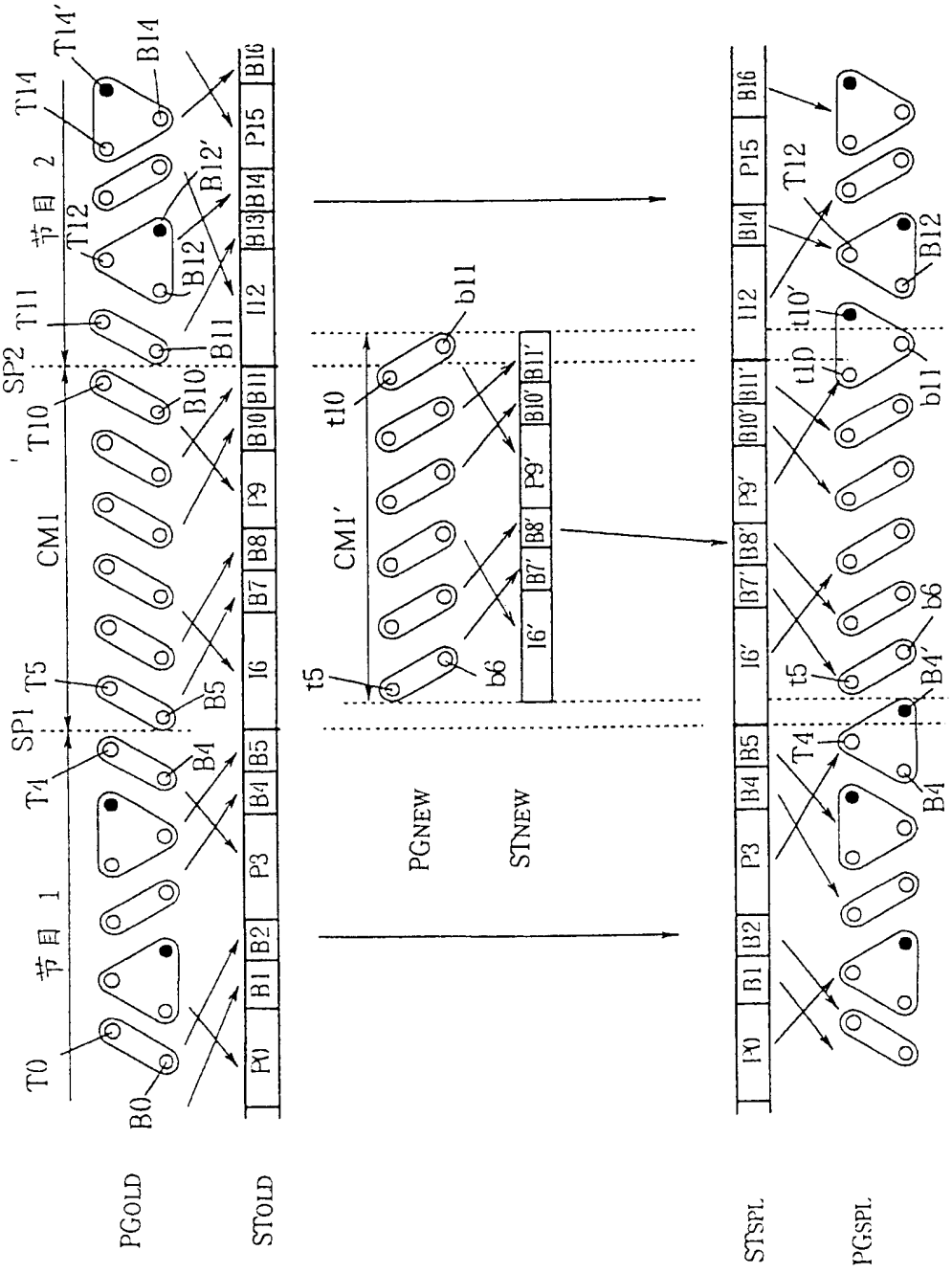


图 22A 图 22B 图 22C

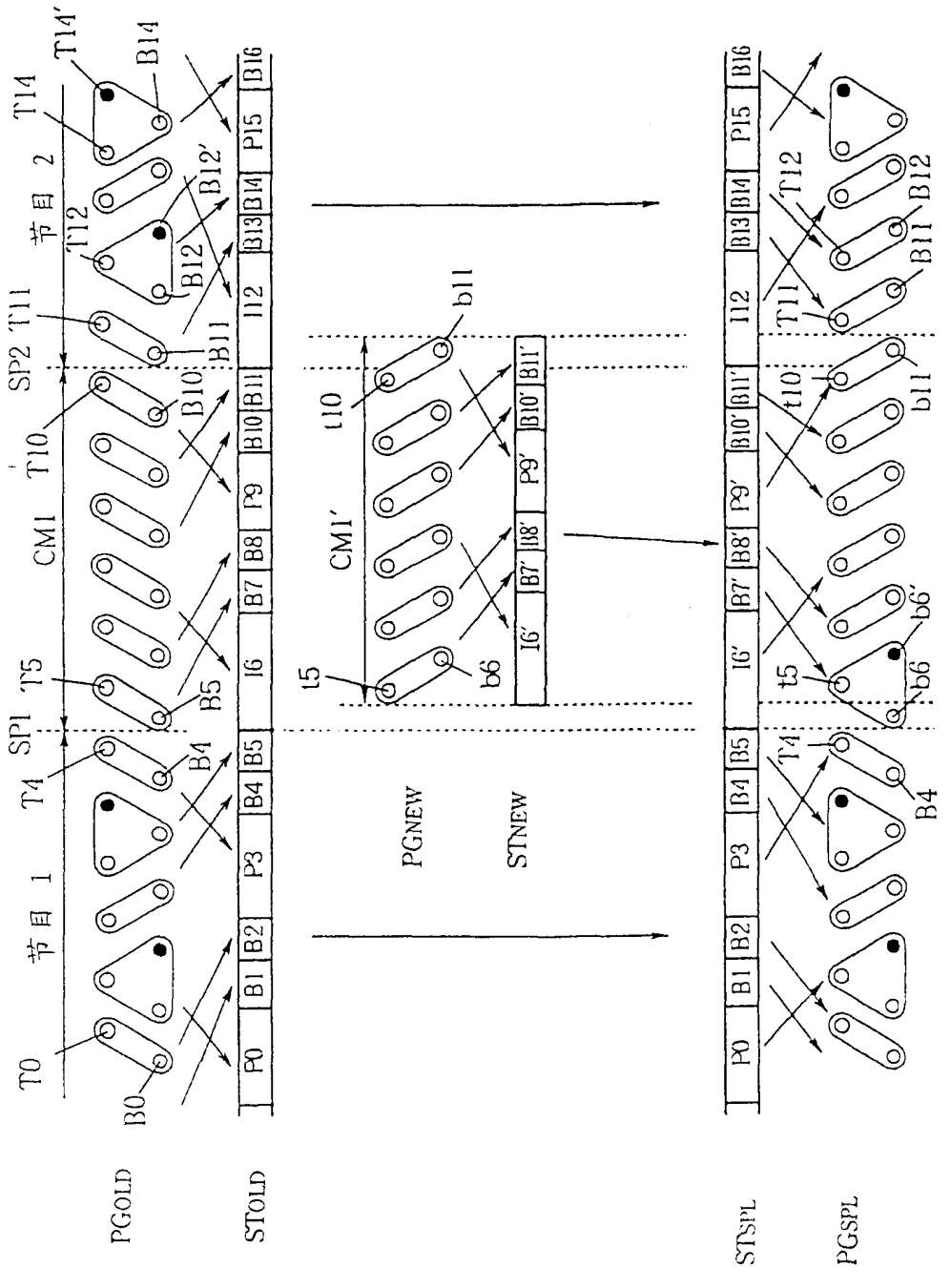


图 23A 图 23B 图 23C