[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
H04N 7/24 (2006.01)
H04N 7/26 (2006.01)

[12] 发明专利说明书

专利号 ZL 01819413.3

[45] 授权公告日 2009 年 2 月 4 日

[11] 授权公告号 CN 100459705C

[22] 申请日 2001.9.4 [21] 申请号 01819413.3

[30] 优先权

[32] 2000.9.25 [33] US [31] 09/669，517

[86] 国际申请 PCT/US2001/027243 2001.9.4

[87] 国际公布 WO2002/028108 英 2002.4.4

[85] 进入国家阶段日期 2003.5.23

[73] 专利权人 通用仪器公司

地址 美国宾夕法尼亚州

[72] 发明人 胡少伟 R·S·罗伯特 V·刘 A·卢沙

[56] 参考文献

WO0041398A1 2000.7.13

US5805220A 1998.9.8

审查员 邹文婷

[74] 专利代理机构 北京纪凯知识产权代理有限公司

代理人 戈泊程伟

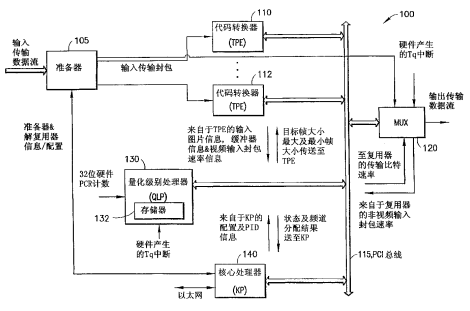
权利要求书 8 页 说明书 30 页 附图 6 页

[54] 发明名称

处理接收多个频道的统计再复用器中数据的方法及其装置

[57] 摘要

本发明揭示一种用来处理包含视频数据的数个频道的高效率统计再复用器。 在一项观点中，当从该数据采集统计信息时，会延迟视频数据的代码转换。 数据的位速率需求参数依据该统计信息来决定，并且视频数据的代码转换依据延迟之后的各自位速率需求参数。 在另一项观点中，于连续时间间隔的多个时段期间会更新视频帧的代码转换位速率，以允许更密切监控位速率。 于每段时间间隔也会更新代码转换位速率的最小及最大界限，例如，缓冲器溢位及下溢保护。 在另一项观点中，会按比例换算帧中宏区块的预先代码转换量化比例(quantization scale)，以依据帧中的预先代码转换数据量与该帧的目标后置代码转换数据量的比例，提供对应的新代码转换量化比例。 随着代码转换的进展，会针对帧的不同部份来调整量化比例，以确保将最低代码转换带宽量配置给每个宏区块。



1. 一种用以处理一接收编码数据的多个频道的统计再复用器中数据的方法，该方法包括下列步骤：

自这些编码数据重新获取视频帧；

当自这些视频帧获取统计信息时，将这些视频帧储存于预览缓冲器中，用以延迟这些视频帧的代码转换；

通过使用复杂度的一个或多个临时平均值和非线性函数，依据自这些视频帧的获取统计信息，决定这些视频帧的各自位速率需求参数；以及

在延迟这些视频帧之后，依据各自位速率需求参数，代码转换各自的这些视频帧。

2. 如权利要求1所述的方法，该方法进一步包括下列步骤：

于一储存装置中储存各自的位速率需求参数；以及

自该储存装置重新获取各自视频帧的位速率需求参数，以在代码转换这些视频帧时使用。

3. 如权利要求1所述的方法，其中：

用来决定各自位速率需求参数的各自视频帧的统计信息包括各自视频帧的平均量化比例值。

4. 如权利要求1所述的方法，其中：

用来决定各自位速率需求参数的各自视频帧的统计信息包括各自视频帧中的位数量。

5. 如权利要求1所述的方法，其中：

用来决定各自位速率需求参数的各自视频帧的统计信息包括与各自视频帧关联的平均位速率。

6. 如权利要求1所述的方法，其中：

用来决定各自位速率需求参数的各自视频帧的统计信息包括各自视频帧中的宏区块数量。

7. 如权利要求1所述的方法，其中：

用来决定各自位速率需求参数的各自视频帧的统计信息包括各自视频帧的宏区块分辨率。

8. 一种接收编码数据的多个频道的统计再复用器，该统计再复用器包括：

装置，用以自这些编码数据重新获取视频帧；

装置，用以当自这些视频帧获取统计信息时，存储这些视频帧用以延迟这些视频帧的代码转换；

装置，用以通过使用复杂度的一个或多个临时平均值和非线性函数，依据自这些视频帧的获取统计信息，决定这些视频帧的各自位速率需求参数；以及

装置，用以在延迟这些视频帧之后，依据各自位速率需求参数，代码转换各自的这些视频帧。

9. 一种用以处理一接收包含编码视频帧的多个频道的统计再复用器中数据的方法，该方法包括下列步骤：

在代码转换一特定视频帧的过程中，通过使用复杂度的一个或多个临时平均值和非线性函数，以连续时间间隔的多个时段，更新该特定视频帧的代码转换位速率；

通过于每个连续时间间隔也会更新的最小及最大级别的至少一级别来限制该更新的代码转换位速率，以提供对应的限制性及更新代码转换位速率；

针对于连续时间间隔期间代码转换该特定视频帧的对应部份，配置该限制性及更新代码转换位速率；

计算该特定视频帧的目标帧大小，用以指示代码转换该特定视频帧所产生的预期的数据量；其中：

通过连续时间间隔中更新的最小及最大预测值的至少一个来限制该目标帧大小；

根据该目标帧大小决定连续时间间隔中该特定视频帧的代码转换位速率。

10. 如权利要求9所述的方法，其中：

该连续时间间隔期间为周期性的。

11. 如权利要求9所述的方法，其中：

通过于每个连续时间间隔更新的最小及最大级别来限制该更新的代码转换位速率，以提供对应的限制性及更新代码转换位速率。

12. 如权利要求9所述的方法，其中与这些视频帧的关联目标帧大小大于这些视频帧略过代码转换的预先代码转换位数量。

13. 如权利要求9所述的方法，该方法进一步包括下列步骤：

为多个频道的通过数据配置可变传输带宽量；以及

依据该配置步骤，调整一传输带宽量，以供代码转换多个频道的视频帧使用。

14. 如权利要求9所述的方法，其中：

该统计再复用器输出一包含多个统计再复用频道群组的传输数据流；以及

一总可用传输带宽的各自部份用来设定这些统计再复用器群组的各自统计再复用器群组。

15. 如权利要求9所述的方法，该方法进一步包括下列步骤：

依据一与关联代码转换引擎相关的延迟，于这些连续时间间隔延迟该更新及限制性代码转换位速率；以及

在延迟该更新及限制性代码转换位速率之后，依据该更新及限制性代码转换位速率配置传输位速率，以供在代码转换特定视频帧之后传输该特定视频帧时使用。

16. 如权利要求15所述的方法，该方法进一步包括下列步骤：

在代码转换该特定视频帧之后，将该特定视频帧的封包数量提供给一复用器，用以依据该配置的传输位速率，与该频道的至少一其它频道的封包一起复用传输。

17. 一种用以处理一包括编码视频帧的多个频道的统计再复用器

中数据的方法，该方法包括下列步骤：

在代码转换一特定视频帧的过程中，通过使用复杂度的一个或多个临时平均值和非线性函数，以连续时间间隔的多个时段，至少更新该特定视频帧的代码转换位速率；

通过于每个连续时间间隔也会更新的最小及最大级别的至少一级别来限制该更新的代码转换位速率，以提供对应的限制性及更新代码转换位速率；

针对于连续时间间隔期间代码转换该特定视频帧的对应部分，配置该限制性及更新代码转换位速率；以及

计算每个视频帧的目标帧大小，用以指示代码转换该视频帧所产生的预期的数据量；

其中与这些视频帧的关联目标帧大小在与这些视频帧略过代码转换的预先代码转换位数量的预先决定差值范围内。

18. 如权利要求17所述的方法，其中在连续时间间隔中该特定视频帧的代码转换位速率依据该目标帧大小来决定。

19. 如权利要求18所述的方法，其中：

该目标帧大小受到于连续时间间隔更新的最小预测值及最大预测值的至少一值的限制。

20. 如权利要求19所述的方法，其中：

这些最小值及最大值的至少一值依据关联代码转换引擎缓冲器的当前全满所决定。

21. 如权利要求19所述的方法，其中：

决定这些最小值及最大值的至少一值，以防止关联的译码器缓冲器下溢或溢位。

22. 如权利要求18所述的方法，该方法进一步包括下列步骤：

依据该目标帧大小，评估传输包含该特定视频帧的代码转换数据的至少一封包所需的时间。

23. 一种用以处理一包括编码视频帧的多个频道的统计再复用

器中数据的方法，该方法包括下列步骤：

在代码转换一特定视频帧的过程中，通过使用复杂度的一个或多个临时平均值和非线性函数，以连续时间间隔的多个时段，至少更新该特定视频帧的代码转换位速率；

通过于每个连续时间间隔也会更新的最小及最大级别的至少一级别来限制该更新的代码转换位速率，以提供对应的限制性及更新代码转换位速率；

针对于连续时间间隔期间代码转换该特定视频帧的对应部分，配置该限制性及更新代码转换位速率；以及

计算每个特定视频帧的目标帧大小，用以指示代码转换该特定视频帧所产生的预期的数据量；以及

依据该目标帧大小，评估将时钟参考数据播入包含该特定视频帧的代码转换数据的至少一封包所需的时间；

其中根据该目标帧大小决定连续时间间隔中该特定视频帧的代码转换位速率。

24. 如权利要求23所述的方法，其中：

该目标帧大小受到于连续时间间隔更新的最小值及最大值的至少一值的限制。

25. 一种接收包括编码视频帧的多个频道的统计再复用器，该统计再复用器包括：

装置，用以在代码转换一特定视频帧的过程中，通过使用复杂度的一个或多个临时平均值和非线性函数，以连续时间间隔的多个时段，更新该特定视频帧的代码转换位速率；

装置，用以通过于每个连续时间间隔也会更新的最小及最大级别的至少一级别来限制该更新的代码转换位速率，以提供对应的限制及更新代码转换位速率；

装置，用以针对于连续时间间隔期间代码转换该特定视频帧的对应部份，配置该限制性及更新代码转换位速率；

装置，用以计算该特定视频帧的目标帧大小，用以指示代码转换该特定视频帧所产生的预期的数据量；以及

装置，用以依据目标帧大小，评估将时钟参考数据插入包含该特定视频帧的代码转换数据的至少一封包所需的时间；

其中根据该目标帧大小决定连续时间间隔中该特定视频帧的代码转换位速率。

26. 一种用以处理用来接收包括编码视频帧的多个频道的统计再复用器中的数据的方法，其中这些视频帧包含一特定视频帧，在该特定视频帧的关联部份中包括多个宏区块，当将该特定视频帧输入至该统计再复用器时，每个宏块均具有一关联的旧量化比例，该方法包括下列步骤：

计算代码转换该特定视频帧所产生的预期的目标数据量；以及

依据下列条件来决定新的量化比例，以在代码转换该特定视频帧中一第一部份中的对应宏区块时使用：(a)对应的旧量化比例，以及(b)该特定视频帧中预先代码转换数据量与(c)该目标数据量的比例；以及

通过使用复杂度的一个或多个临时平均值和非线性函数，依据统计信息，决定这些视频帧在代码转换中的各自位速率需求参数。

27. 如权利要求26所述的方法，该方法进一步包括下列步骤：

将新量化比例调整为一的整数。

28. 如权利要求26所述的方法，其中：

这些部份包含该特定视频帧的各自切片(slice)。

29如权利要求26所述的方法，其中：

若需要，将这些新量化比例调整为比对应的旧量化比例更粗略。

30. 如权利要求26所述的方法，其中：

若需要，调整这些新量化比例，以确保对应宏区块接受代码转换这些宏区块的最小位数量。

31. 如权利要求26所述的方法，该方法进一步包括下列步骤：

依据下列条件来决定新的量化比例，以在代码转换该特定视频帧中一第一部份中的宏区块时使用：(a)对应的旧量化比例，以及(b)该特定视频帧中预先代码转换数据量与(c)代码转换这些剩余部份所产生的

预期的目标数据量的比例。

32. 如权利要求31所述的方法，其中：

决定代码转换这些剩余部份所产生的预期目标数据量的方式为，将代码转换该特定视频帧所产生的预期目标数据量扣减代码转换该第一部份所产生的数据量。

33. 如权利要求31所述的方法，其中：

决定这些剩余部份中的预先代码转换数据量的方式为，将该特定视频帧中的预先代码转换数据量扣减该第一部份中的预先代码转换数据量。

34. 如权利要求26所述的方法，该方法进一步包括下列步骤：

维护一数据量运行计数，以供代码转换这些部份时使用；

其中，如果运行计算超过一最大级别，则会设定一应急量化比例，直到留下足够的位供剩余部份中的宏区块使用，以接受代码转换这些宏区块的最小位数量。

35. 如权利要求34所述的方法，其中：

在代码转换每个部份之后，会将最大级别向下调整。

36. 如权利要求26所述的方法，其中：

每个宏区块都具有自己关联的旧量化比例。

37. 如权利要求26所述的方法，其中：

一组宏区块群组都具有自己关联的旧量化比例。

38. 一种接收包括编码视频帧的多个频道的统计再复用器，其中这些视频帧包含一特定视频帧，在该特定视频帧的关联部份中包括多个宏区块，当将该特定视频帧输入至该统计再复用器时，每个宏区块均具有一关联的旧量化比例，该统计再复用器包括：

装置，用以计算代码转换该特定视频帧所产生的预期的目标数据量；以及

装置，用以依据下列条件来决定新的量化比例，以在代码转换该

特定视频帧中一第一部份中的对应宏区块时使用：(a)对应的旧量化比例，以及(b)该特定视频帧中预先代码转换数据量与(c)该目标数据量的比例；以及

装置，通过使用复杂度的一个或多个临时平均值和非线性函数，依据自这些视频帧的统计信息，决定这些视频帧的各自位速率需求参数。

处理接收多个频道的统计再复用器中数据的方法及其装置

发明背景

本发明与一种用来代码转换数字视频信号的统计再复用器有关。

通常需要调整提供给(例如)有线电视网络等等的用户终端机的数字视频节目的位速率。例如，可经由卫星传输于头端接收第一群信号。当增加节目(例如商业或其它内容)时，头端业者可能想要将所选节目从本地来源(如储存媒体或本地现场转播)发送给用户。此外，通常需要在整个可用的频道带宽内提供节目。

因此，已发展出统计再复用器(缩写为：stat remux)或多频道代码转换器，其以特定位速率重新压缩已经过预先压缩的视频比特流的方式来处理已经过预先压缩的视频比特流。

在此类系统中，由并联排列的处理器来处理数个频道。每个处理器通常都可处理多重频道数据。然而，在某些情况下(如需要许多计算的HDTV)，可能会在多重处理器之中配置单一频道的数据部份。另外，通常会将固定代码转换带宽配置给一组或一组以上频道(统计再复用器群组)。

但是，需要有一种改良的统计再复用系统，其提供每个频道的位速率需求参数，使针对代码转换频道来配置位的原则为，最佳化已编码数据的影像品质，并且仍然符合有限总处理能力的限制条件。

系统应利用自比特流所导出的统计信息来评估位速率需求参数，如帧位计数及原始比特流的平均量化比例值。系统应兼容于MPEG-2比特流。系统应依据利用原始比特流的统计信息所评估的编码复杂度，以配置I、P及B帧的目标输出帧位计数。

另外，系统应提供帧内的MPEG-2宏区块处理，其方式是利用宏区块位计数及原始比特流的平均量化比例值以支配传输率控制处理，以符合输出的目标帧位计数。

在视频帧中的一些时间，系统提供配置代码转换位速率的周期调整。

此外，系统应依据原始、预先代码转换量化比例值，导出用来代

此外，系统应依据原始、预先代码转换量化比例值，导出用来代码转换帧中宏区块的量化比例值。随着帧的代码转换进展，应调整量化比例值，以确保以将最少数量的位配置给要代码转换的每个宏区块。

本发明提供一种具有前述及其它优点的系统。

发明内容

本发明与一种用来代码转换数字视频信号的统计再复用器有关。

在本发明的一项观点中，会利用帧位计数及原始比特流(如MPEG-2比特流)的平均量化比例值(整个帧的平均值)所导出的统计信息来评估位速率需求参数。提供(例如)五帧的预览。

本发明在代码转换频道之中配置总可用带宽。

本发明依据利用帧位计数及原始比特流的平均量化比例值(整个每一输入帧的平均值)所评估的编码复杂度，以配置I、P及B帧的输出帧位计数目标。

另外，在本发明的另一项观点中，于帧内的MPEG-2宏区块处理期间，会使用宏区块位计数及原始比特流的平均量化比例值来支配传输率控制处理，以符合输出的目标帧位计数。

因此，本发明揭示一种用来处理包含视频数据的数个频道的高效率统计再复用器。在本发明的一项观点中，当从该数据采集统计信息时，会延迟视频数据的代码转换。数据的位速率需求参数依据该统计信息来决定，并且视频数据的代码转换依据延迟之后的各自位速率需求参数。

在本发明的另一项观点中，于连续时间间隔的多个时段期间会在统计再复用器更新视频帧的代码转换位速率，以允许更密切监控位速率。另外，于每段时间间隔会更新代码转换位速率的最小及最大界限。因此，在第一时间间隔会代码转换帧的一部份，然后更新代码转换位速率，接着在第二时间间隔会代码转换帧的第二部份，然后更新代码转换位速率，以此类推。

在本发明还有一项观点中，会按比例换算帧中宏区块的预先代码转换量化度量(quantization scale)，以依据帧中的预先代码转换数据量

与该帧的目标后置代码转换数据量的比例，提供对应的新代码转换量化度量。另外，随着代码转换的进展，会针对帧的不同部份来调整量化比例，以确保将最低代码转换带宽量配置给每个宏区块。

揭示对应的方法及装置。

本发明提供一种用以处理一接收编码数据的多个频道的统计再复用器中数据的方法，该方法包括下列步骤：自这些编码数据重新获取视频帧；当自这些视频帧获取统计信息时，将这些视频帧储存于预览缓冲器中，用以延迟这些视频帧的代码转换；通过使用复杂度的一个或多个临时平均值和非线性函数，依据自这些视频帧的获取统计信息，决定这些视频帧的各自位速率需求参数；以及在延迟这些视频帧之后，依据各自位速率需求参数，代码转换各自的这些视频帧。

本发明提供一种接收编码数据的多个频道的统计再复用器，该统计再复用器包括：装置，用以自这些编码数据重新获取视频帧；装置，用以当自这些视频帧获取统计信息时，存储这些视频帧用以延迟这些视频帧的代码转换；装置，用以通过使用复杂度的一个或多个临时平均值和非线性函数，依据自这些视频帧的获取统计信息，决定这些视频帧的各自位速率需求参数；以及装置，用以在延迟这些视频帧之后，依据各自位速率需求参数，代码转换各自的这些视频帧。

本发明提供一种用以处理一接收包含编码视频帧的多个频道的统计再复用器中数据的方法，该方法包括下列步骤：在代码转换一特定视频帧的过程中，通过使用复杂度的一个或多个临时平均值和非线性函数，以连续时间间隔的多个时段，更新该特定视频帧的代码转换位速率；通过于每个连续时间间隔也会更新的最小及最大级别的至少一级别来限制该更新的代码转换位速率，以提供对应的限制性及更新代码转换位速率；针对于连续时间间隔期间代码转换该特定视频帧的对应部份，配置该限制性及更新代码转换位速率；计算该特定视频帧的目标帧大小，用以指示代码转换该特定视频帧所产生的预期的数据量；其中：通过连续时间间隔中更新的最小及最大预测值的至少一个来限制该目标帧大小；根据该目标帧大小决定连续时间间隔中该特定视频帧的代码转换位速率。

本发明提供一种用以处理一包括编码视频帧的多个频道的统计再复用器中数据的方法，该方法包括下列步骤：在代码转换一特定视频帧的过程中，通过使用复杂度的一个或多个临时平均值和非线性函数，以连续时间间隔的多个时段，至少更新该特定视频帧的代码转换位速率；通过于每个连续时间间隔也会更新的最小及最大级别的至少一级别来限制该更新的代码转换位速率，以提供对应的限制性及更新代码转换位速率；针对于连续时间间隔期间代码转换该特定视频帧的对应部分，配置该限制性及更新代码转换位速率；以及计算每个视频帧的目标帧大小，用以指示代码转换该视频帧所产生的预期的数据量；其中与这些视频帧的关联目标帧大小在与这些视频帧略过代码转换的预先代码转换位数量的预先决定差值范围内。本发明提供一种用以处理一包括编码视频帧的多个频道的统计再复用器中数据的方法，该方法包括下列步骤：在代码转换一特定视频帧的过程中，通过使用复杂度的一个或多个临时平均值和非线性函数，以连续时间间隔的多个时段，至少更新该特定视频帧的代码转换位速率；通过于每个连续时间间隔也会更新的最小及最大级别的至少一级别来限制该更新的代码转换位速率，以提供对应的限制性及更新代码转换位速率；针对于连续时间间隔期间代码转换该特定视频帧的对应部分，配置该限制性及更新代码转换位速率；以及计算每个特定视频帧的目标帧大小，用以指示代码转换该特定视频帧所产生的预期的数据量；以及依据该目标帧大小，评估将时钟参考数据播入包含该特定视频帧的代码转换数据的至少一封包所需的时间；其中根据该目标帧大小决定连续时间间隔中该特定视频帧的代码转换位速率

本发明提供一种接收包括编码视频帧的多个频道的统计再复用器，该统计再复用器包括：装置，用以在代码转换一特定视频帧的过程中，通过使用复杂度的一个或多个临时平均值和非线性函数，以连续时间间隔的多个时段，更新该特定视频帧的代码转换位速率；装置，用以通过于每个连续时间间隔也会更新的最小及最大级别的至少一级别来限制该更新的代码转换位速率，以提供对应的限制及更新代码转换位速率；以及装置，用以针对于连续时间间隔期间代码转换该特定

视频帧的对应部份，配置该限制性及更新代码转换位速率；装置，用以计算该特定视频帧的目标帧大小，用以指示代码转换该特定视频帧所产生的预期的数据量；其中：通过连续时间间隔中更新的最小及最大预测值的至少一个来限制该目标帧大小；根据该目标帧大小决定连续时间间隔中该特定视频帧的代码转换位速率。

本发明提供一种用以处理用来接收包括编码视频帧的多个频道的统计再复用器中的数据的方法，其中这些视频帧包含一特定视频帧，在该特定视频帧的关联部份中包括多个宏区块，当将该特定视频帧输入至该统计再复用器时，每个宏块均具有一关联的旧量化比例，该方法包括下列步骤：计算代码转换该特定视频帧所产生的预期的目标数据量；以及依据下列条件来决定新的量化比例，以在代码转换该特定视频帧中一第一部份中的对应宏区块时使用：(a)对应的旧量化比例，以及(b)该特定视频帧中预先代码转换数据量与(c)该目标数据量的比例；以及通过使用复杂度的一个或多个临时平均值和非线性函数，依据统计信息，决定这些视频帧在代码转换中的各自位速率需求参数。

本发明提供一种接收包括编码视频帧的多个频道的统计再复用器，其中这些视频帧包含一特定视频帧，在该特定视频帧的关联部份中包括多个宏区块，当将该特定视频帧输入至该统计再复用器时，每个宏区块均具有一关联的旧量化比例，该统计再复用器包括：装置，用以计算代码转换该特定视频帧所产生的预期的目标数据量；以及装置，用以依据下列条件来决定新的量化比例，以在代码转换该特定视频帧中一第一部份中的对应宏区块时使用：(a)对应的旧量化比例，以及(b)该特定视频帧中预先代码转换数据量与(c)该目标数据量的比例；以及装置，通过使用复杂度的一个或多个临时平均值和非线性函数，依据自这些视频帧的统计信息，决定这些视频帧的各自位速率需求参数。

附图说明

图1显示根据本发明的统计再复用，以及数据流入及流出量化级别处理器(quantization level processor；QLP)。

图2显示用来根据本发明使用的简化型代码转换器的附图。

图3显示用来根据本发明使用的简化型代码转换器的附图，用以执行重新量化，而不需要进行运动补偿。

图4显示根据本发明的端对端统计再复用器处理延迟的附图。

图5显示根据本发明的代码转换器视频缓冲核验器(video buffering verifier；VBV)模型简化型代码转换器的附图。

图6显示根据本发明的代码转换器传输率时序的附图。

图7显示根据本发明QLP与代码转换器处理组件(transcoder processing element；TPE)之间通信时序的附图。

具体实施方式

本发明与一种用来代码转换数字视频信号的统计再复用器有关。下列是使用的缩写与术语：

BW–带宽(Bandwidth)

DCT–离散余弦转换(Discrete Cosine Transform)

DTS–译码时戳(Decoding Time Stamp)

ES–基本数据流(Elementary Stream)

FIFO–先进先出(First-In, First-Out)

KP–核心处理器(Kernel Processor)

MTS–MPEG传输数据流(MPEG Transport Stream)

PCI–周边组件互连(Peripheral Component Interconnect)

PCR–节目时钟参考(Program Clock Reference)

PES–封包化基本数据流(Packetized Elementary Stream)

PID–节目识别项(Program Identifier)

Q–量化(Quantization)

QLP–量化级别处理器(Quantization Level Processor)

SDRAM–静态动态随机存取存储器(Static Dynamic Random Access Memory)

TP–传输封包(Transport Packet)

TPE–代码转换器核心处理组件(Transcoder core Processing Element)

VLD–可变长度型译码(Variable-Length Decoding)

VLE–可变长度型编码(Variable-Length Encoding)

图1显示根据本发明的统计再复用，以及数据流入及流出量化级别处理器(quantization level processor；QLP)。

统计再复用器100包括准备器(groomer)105，用来接收一些输入运输数据流。不同视频服务中的对应输入运输数据流会提供数个代码转换器110, ..., 112,或TPE (transcoding engine；代码转换引擎)的其中一个代码转换器。通常每个代码转换器均可处理一个或一个以上视频服务(频道)。经代码转换的数据会经由PCI总线115提供给复用器(mux)120，其负责组合对应的输出传输数据流。

核心处理器(KP)设定准备器105、TPE 110, ..., 112、QLP 130及Mux 120。

具体而言，Mux 120响应QLP 130(其具有如SDRAM之类的存储器132)所提供的传输位速率。QLP可使用媒体处理器实施，如Equator Technologies公司销售的MAPCA2000 (300MHz)。QLP 130执行下列功能：

- 将可用的带宽配置给输出视频服务，以最佳化视频品质，并决定要代码转换的每个帧的目标帧大小。
- 接收来自于核心处理器的配置设定参数。
- 向核心处理器报告运作状态及统计数据。

QLP经由PCI总线(32位@ 66MHz)来与KP和TPE通信。配置QLP的SDRAM上的存储器区块，以供处理器间通信使用。QLP与其它处理器「共享」这个存储器区块。

输入至QLP 130

- 配置设定参数及命令(来源：KP 140)
- 视频与关联的音讯和数据输入封包传输率信息(来源：TPE 110, ..., 112)
- 要代码转换的输入帧的统计数据(来源：TPE)
- 输入帧的时序信息(来源：TPE)
- 刚才代码转换的输出帧的统计数据(来源：TPE)
- 输出帧的时序信息(来源：TPE)
- 代码转换器FIFO级别(来源：TPE)
- 非视频(数据)输入封包传输率信息(来源：Mux 120)

自QLP 130 输出

- 状态及统计数据(目的地：KP)
- TPE服务指派信息(目的地：KP)
- 传输位速率(目的地：Mux)
- 代码转换目标帧大小(目的地：TPE)
- 缓冲器保护的最大及最小帧大小(目的地：TPE)
- 要插入帧中的最少PCR数量(目的地：TPE)
- 命令TPE通过帧的旗标(目的地：TPE)

1. 概览

虽然代码转换器100不一定要译码及重新编码视频数据流，但是代码转换功能可仿真全译码及重新编码。下列步骤概述根据本发明的传输率控制及统计再复用系统。下一部份会说明细节。

1. 每个TPE 110, ..., 112 输入其正在处理的所有视频频道的传输数据流。然后，将传输数据流解封包化，并且仿真视频译码器缓冲器。每个TPE均使用一个预览缓冲器来储存数个未来的帧，并且从这些帧获得统计信息。具体而言，针对要代码转换的所有输入帧，TPE计算平均量化比例值及输入帧中的位数量。在输入帧之后被代码转换之前的至少1.5 NTSC帧周期(frame time)的排定时间，QLP 130利用这些参数来计算输入帧的位速率需求参数。具体而言，因为视频频道中可能使用3：2折叠式(pulldown)格式，所以已编码帧可能是一个NTSC帧周期(33.3ms)或1.5帧周期(50ms)。此处使用较长的时间，以确保能

够在实际代码转换开始之前，决定出帧的代码转换速率配置。

2. QLP 130执行带宽配置处理程序，以便以周期时间间隔T_q将代码转换位速率配置给TPE。在每个Tq时间间隔，QLP 130计算所有视频频道的代码转换位速率。代码转换位速率储存在QLP 130的队列中，并且延迟(0.5秒+3 NTSC帧周期)=0.6秒，调整为最近的Tq周期。一个NTSC帧周期是1/30秒。视频频道的代码转换位速率的延迟值变成传输位速率，并且供Mux 120使用。

3. 虽然帧位于TPE的预览缓冲器中，但是会使用整个帧的平均代码转换位速率，以导出目标帧大小的初始值，这是代码转换之后的帧预测大小。初始目标帧大小值储存在QLP 130的输出帧大小队列中(例如，储存在存储器132中)，并且当关联的TPE准备要代码转换帧时撷取该值。QLP可在存储器132中实施队列。

4. 当代码转换器准备要代码转换新帧时，会从输出帧大小队列撷取先前决定的初始目标帧大小值。依据代码转换器缓冲器的当前状态(全满)，会计算出用来界定初始目标帧大小的最大和最小帧大小，以防止译码器缓冲器下溢或溢满。

5. 如果目标位数量(目标帧大小)大于(或接近、在预先决定容限内–请参阅第6.3节)输入帧中的位数，由于代码转换的用途是降低帧中的位数量，所以这个帧以通过模式绕过代码转换。当输入比特流已经过紧密压缩时，则会发生这个情况。当略过一个帧时，会重新封包化关联的输入基本数据流。如果目标位数量小于输入帧中的位数，则帧会经过位缩减过程(代码转换)。例如，可透过简化的代码转换器架构(图2)或重新量化(图3)来执行位缩减。

6. 依据每帧的目标位数量及原量化比例选取要代码转换的每个宏区块的量化比例。必须符合输出量化比例高于(较粗略)输入量化比例的条件。

7. 于代码转换期间，在TPE输出的封包中，TPE必须配置PCR字段的某些时隙。避免配置超过需要的时隙非常重要，因为这会浪费位。因此，外传封包在TPE上建立并储存于存储器中，例如，TPE FIFO缓冲器中。另外，QLP 130使用目标帧大小来评估传输帧所使用的时间，并由此评估插入PCR的时间。

另外，至少每隔0.1秒建立PCR时隙，以符合MPEG2系统标准的需求。

8. 于每个n*Tq期间，Mux 120经由PCI总线115从TPE 110, ..., 112读取指派给每个频道的封包数量。「n」是Mux 120的设计参数，并且可能是任何正整数。这个封包指派相当于传输位速率配置，接着是代码转换位速率配置的延迟版本。也就是说，会将位速率转换成每Tq期间传送至Mux的封包数量。

9. Mux 120 每隔27MHz时钟的m个刻度(tick)接收一个传输刻度。如果传输的封包含有PCR，则Mux会执行PCR校正，以提供与代码转换器主时钟适当同步的PCR值。如R. Nemiroff, V. Liu与S. Wu, 于20009月22日提出的共同转让、共同申请美国专利申请案号09/667,734，标题为「Regeneration Of Program Clock Reference Data For Mpeg Transport Streams」中说明，可实现这项功能。传输封包经由PCI总线115自Mux处理器送出。传输刻度表示输入一个传输封包的时序时间间隔。

图2显示用来根据本发明使用的简化型代码转换器的附图。

虽然简单的代码转换器可能仅仅是串联的MPEG译码器及编码器，但是代码转换器200提供减少计算的简化设计。代码转换器架构200执行DCT领域中的大部份作业，所以可减少反相-DCT及运动补偿作业的数量。另外，由于不会重新计算运动向量，所以会显著减少所需的计算。这个简化的架构提供低计算复杂度及高弹性的理想组合。

具体而言，将预先压缩视频比特流输入至可变长度型译码器(Variable Length Decoder(VLD)215.解量化天能(反向量化器)220使用第一量化级大小Q_1来处理VLD 215的输出。

将运动向量(MV)数据从VLD 215提供给运动补偿功能235，以响应像素领域数据的前一帧缓冲器250及/或当前帧缓冲器245。DCT功能270将MC功能235的输出转换成频率领域，并将结果提供给加法器230。开关231将加法器230或Q_1^{-1}功能220的输出传送至量化功能Q_2 275，其量化数据，通常以较粗略等级，以降低位速率。然后，在反向量化功能Q_2^{-1} 282将输出反向量化，以在加法器286与开关231的输出相加。将在加法器286的输出提供给IDCT功能284，并将其输出提

供给帧缓冲器245和250。

可变长度编码器(Variable Length Encoder；VLE)280编码量化功能275的输出，以降低的位速率提供输出比特流。以此方式，通过变更Q_2来调整代码转换器的位输出速度。

图3显示用来根据本发明使用的简化型代码转换器300的附图，用以执行重新量化，而不需要进行运动补偿。

此处，帧只会经过重新量化处理，而不会运动补偿。一般而言，避免IDCT和DCT作业。这个策略能够降低复杂度，但是需要对输出数据进行一些人工处理。DCT系数被解量化，然后重新量化。

具体而言，使用VLD 410、反向量化器420、量化器430及VLE 440。

2. 端对端处理延迟

图4显示根据本发明的端对端统计再复用器处理延迟的附图。

代码转换器或TPE 110的实例包括用以缓冲输入传输数据流的MTS缓冲器405、用以离析传输数据流中不同服务的基本数据流的解复用器410以及用以储存ES数据流的ES缓冲器 415。在VLD功能420将ES数据经过可变长度译码，并将结果提供给具有(例如)五个帧容量的预览延迟缓冲器425。于缓冲器435延迟一个帧之后，在代码转换功能440将帧经过代码转换，并将结果储存于代码转换缓冲器445中。重新复用器(remux)450组合来自于代码转换缓冲器445的数据与(若有)来自于传输数据流延迟缓冲器430的数据，并且将产生的传输数据流传送至译码器452，如宽频通信网路中的视频转接器(set-top box)。传输数据流延迟缓冲器430用来略过不经过代码转换的帧，如上文所述。略过的帧会被延迟，以维持与被代码转换的其它频道同步。

请注意，实际上，会将来自于代码转换器110的输出数据流与来自于其它代码转换器的传输数据流组合在一起，以构成要传送至典型译码器452的传输复用。译码器452包括用以缓冲传入数据的FIFO缓冲器455，以及用以译码数据的译码功能460，以提供(例如)要显示在电视上的输出。

例如，视频封包会经过0.5秒的缓冲器延迟，这可视不同实施而异。这是介于代码转换(编码)时间与译码时间之间的延迟。代码转换

器(输出)视频FIFO 445及译码器FIFO 455中都会发生这个延迟。缓冲器延迟是固定值。如果延迟代码转换时间，则会缩短译码时间的实际代码转换时间，但是缓冲器延迟会使译码时间的代码转换「刻度」维持固定。

3. 缓冲器模型

图5显示根据本发明的代码转换器VBV模型的附图。

译码器452的VBV模型用来在代码转换新帧之前，先限制最大及最小帧大小。代码转换器的比特流输出FIFO 的级别可用来导出就在新帧DTS之前的译码缓冲器状态(请参阅图7)。具体而言，会以下列方式来给定未来译码缓冲器状态(vbv_fullness)：

vbv_fullness=(从当前时间到新帧DTS时间所要传输的位数量)-(编码器FIFO中的位数量)。图5显示vbv_fullness计算，其中于500显示代码转换器FIFO的构成，于550显示译码器FIFO的构成。

另外，我们可计算已传输的位数量，其方式是将从t秒到QLP所发出的最后编码速率的所有编码速率相加(其中t是整个编码加译码缓冲器的总延迟，即，系统延迟)。QLP以T_q时间期间要输出的封包数量提供位速率。

另外，由于从QLP发出速率变更时间至新速率在代码转换器上实际生效时间的可变等待时间所造成的不确定性，导致必须加入边缘。新位速率在固定期间T_q变更。T_q与视频帧时间(译码器的DTS)不同步。

如图6所示，于虚线(例如，602, 604, 606,...)计算代码转换速率。这个实例假设系统延迟是三个帧，并且必须计算P1-1至P1-3的传输率。运用这个标记法，P1-1标示节目(比特流#1)帧#1、P1-2标示节目(比特流#1)帧#2，以此类推。由于帧DTS时间未校准于速率变更，所以造成代码转换速率与传输速率之间的差异。另外，由于T_q期间横跨两个帧，所以将第二(后项)帧指派给封包。

代码转换与传输之间最糟的速率错误情况是，于当前时间配置的封包数量与系统延迟时间之后(当前帧的DTS)的封包数量之间的差异。图6的下方显示两种极端的案例。在第一案例(650)中，就在T_q之前发生帧DTS。在第二案例670中，就在T_q^+发生帧DTS。A到AA代表

指派给每个T_q期间的封包数量。这两个案例都具有相同的编码封包指派Sum(B到X)。案例1的传输封包数量是Sum(C, D, E, ... , W, X, Y)；而案例2是 Sum(B, C, D, ... ,V,W,X)。案例2没有编码速率与传输速率之间的差异，所以是最理想的案例(DTS对准T_q)。案例1是最槽的情况，其具有B个封包的差异。

因此，评估从当前时间至DTS时间要传输的位数量是：

Σ transcoding_packets(n)± packet_count_error,

packet_count_error=(于DTS时间指派的封包数量)-(于当前时间指派的封包数量)。正封包计数错误及负封包计数错误会对帧大小计算造成不同的影响。

请注意，系统延迟应是T_q的倍数。

运用下列等式所给定的评估vbv fullness

vbv_fullness=(要传输的位数量)–(代码转换器FIFO中的位数量);

这个数值可用来限制代码转换的帧大小，所以它不会超过vbv_fullness。为了确保当译码当前帧(即，将要被代码转换的帧)时，译码器缓冲器不会下溢，一定要符合这个需求。

可从传输位速率序列、代码转换器缓冲器级别的快照及前一代码转换帧的大小导出最大帧大小及最小帧大小，如下所示：

假设B(t)=于时间t的缓冲器级别。

tc=当前帧进入代码转换器FIFO的时间。

t0=最后读取代码转换器FIFO级别的时间。

T(t)=于时间t进入FIFO的帧数量。

R(t)=于时间t的传输位速率。

dts=当前帧的DTS。

nextDts=下一帧的DTS。

D=译码器缓冲器大小。

$$\text{最大帧大小}=\sum_{t=tc}^{dts} R(t) - B(tc)$$

$$=\sum_{t=t0}^{dts} R(t) - \sum_{t=t0}^{tc} R(t) - \left[B(t0) + \sum_{t=t0}^{tc} T(t) - \sum_{t=t0}^{tc} R(t) \right]$$

$$=\sum_{t=t0}^{dts} R(t) - B(t0) + \sum_{t=t0}^{tc} T(t)$$

$$\text{最小帧大小}= \sum_{t=tc}^{nextDts} R(t) - B(tc) - D$$

$$=\text{最大帧大小}+ \sum_{t=dts}^{nextDts} R(t) - D$$

图7显示根据本发明量化级别处理器(quantization level processor; QLP)与代码转换器处理组件(transcoder processing element; TPE)之间通信时序的附图。

于时间705，TPE将当前帧「N」的统计信息传送至QLP。于时间710，TPE传送关于TPE输出缓冲器全满的信息，其中包括来自于前一代码转换帧的数据及N-k指数，其中k=1。前一编码帧通常是N-1，即，前一帧。但是，有时候代码转换帧可能会花费一个以上帧时间，使得时序有误差。在此情况下，「当前帧」与刚刚代码转换的帧之间的距离可能会超过一个以上帧，例如，k=2。于时间715，使用从关联统计信息所计算而得的需求参数，开始帧「N」的代码转换。针对每个Tq期间计算代码转换位速率，例如于实例时间720。

时间725标示开始下一帧的代码转换，其具有指数N-1。

于实例时间730，TPE将关于其输出缓冲器(现在含有来自于帧N的数据)全满的信息传送给QLP。为了响应，于时间735，QLP将目标帧大小及代码转换位速率的最大及最小界限提供给TPE。

时间740及745分别标示帧N和N+1的译码时戳时间。

于时间750，QLP将传输位速率传送至复用器，以告知复用器TPE的输出缓冲器中有多少数据封包要在传输数据流中输出。时间750经过延迟期间接在代码转换时间720之后。

4. 需求参数

依据预期的帧复杂度，以决定每个帧的位速率需求参数。QLP 130依据该需求参数及可用的带宽，将代码转换位速率配置给每个TPE。

请再次参考图4，可变长度译码器420会先将输入帧位经过局部译

码，并且计算平均量化比例及帧中的位数量。于提供对应预览延迟的预览缓冲器425中储存每个视频频道的局部译码帧数量(例如，五个帧)系数及标题。处理器SDRAM存储器132的大小会限制预览缓冲器的长度。每个系数占用两个字节，结果是720x480x1.5x2= 1 Mbyte/帧。

于目标译码器452依据预定帧译码时间所决定特定时间$T_{frameStart}$，计算预览缓冲器425中最旧帧的需求参数。译码时间由帧的DTS指定，这是27MHz时钟刻度单元。$T_{frameStart}$被定义为：

(帧的译码时间-缓冲器延迟-1.5 NTSC帧时间)。

需求参数从平均量化比例及输入帧的位计数计算而得，如下所示：

NeedParameter = MbResolutionAdjust * AvgQR * (CurrentQR + Alpha * PastQR)/(Beta * CurrentQR + PastQR)

，其中

AvgQR=(在最近15 P或B帧及过去最近I帧期间(avgInQuant * inFrameSize)的总和)*900,000/(当前帧的DTS-过去第16个帧的DTS)。900,000是一个帧期间(1/30秒)中的27 MHz单元数量。27 MHz是MPEG时钟速率。

如果过去(例如)45帧内没有I帧，则会使用过去的16 P或B帧。

针对I帧，

CurrentQR=avgInQuant * 当前帧的inFrameSize。

PastQR=avgInQuant *最后I帧的inFrameSize。如果过去45帧内没有I帧，则会将PastQR设定等于CurrentQR值。

针对P或B帧，

CurrentQR=在当前帧及同一图片类型的预览缓冲器425中所有帧期间(avgInQuant * inFrameSize)的平均值。

PastQR= 同一图片类型的过去四个帧期间(avgInQuant * inFrameSize)的平均值。如果过去同一图片类型的帧少于4个帧，则会将PastQR设定等于CurrentQR值。

Alpha及Beta是可调整的参数，以控制需求参数对量化比例与位计数乘积变化的反应。默认值是Alpha=256、Beta=256。

MbResolutionAdjust是可调整的参数，不同分辨率失真方面的感

知差异。分辨率愈低，愈容易发现失真。因此针对较低的分辨率提高需求参数。全分辨率的MbResolutionAdjust默认值是1.0，四分之三分辨率的默认值是1.2以及半分辨率的默认值是1.5。或者，或此外，可依据宏区块分辨率来调整需求参数，这是帧中的宏区块数量。

5. 输入位速率信息

在所有的Tq时隙中，TPE 110, ..., 112及Mux 120计数输入传输封包的数量，并将这个封包计数信息储存在QLP 130中的环形缓冲器中。TPE 处理的每个视频节目/频道都有一个输入位速率的环形缓冲器，并且会将所有数据流的位速率信息环形缓冲器直接传送至Mux 120，而不需要通常TPE(即，以通过模式)。每个环形缓冲都具有(例如)1024个输入项，并且每个输入项储存一个Tq时隙的位速率信息。1024个输入项仅仅是可针对不同实施改变的设计参数。环形缓冲器的大小应足以保存0.6秒延迟的数据。QLP 130可从封包计数计算出每个Tq时隙的瞬间输入位速率，如下所示：

BitRate(每秒位数量)=PktCount * 188 * 8 / TqPeriod.

Mux 120计算每个数据服务中的传输封包数量(空封包除外)，这可包含一个或一个以上MPEG节目。QLP 130使用封包计数来计算出每个Tq时隙的瞬间数据服务输入位速率。Mux将封包计数储存在QLP上的环形缓冲器中，其方式与储存来自于TPE的封包计数信息一样。

Mux 120和TPEs将封包计数信息写入至QLP环形缓冲器中的处理程序与Tq刻度同步。与封包计数信息一起储存的Tq指数用来于初始化期间，使QLP与封包计数信息同步。

Tq指数由QLP维护。在初始化时，QLP将Tq指数设定为0，并且在每个Tq中断将Tq指数递增1。QLP将Tq指数及关联的时间定期广播至TPE 110, ..., 112 及 Mux 120。

于代码转换位速率配置处理程序期间，QLP 130撤销纯粹通过视频频道及非视频频道的带宽。由于这些通过频道中封包的传输位速率必须符合输入端对应封包的位速率，所以针对每个通过视频频道撤销的位速率等于下列时间的瞬间输入位速率：

PacketCountDelay = PassThroughDelay -TcrToTxrDelay

在当前时间之前，其中PassThroughDelay是通过视频频道中封包

的延迟(从解复用器410至重新复用器450)，在示范性实施中，其固定在(0.5秒+6 NTSC帧期间)=0.7秒。非视频PID具有相同的延迟数量。

TcrToTxrDelay是从代码转换速率计算(当前Tq刻度)至传输位速率实施的延迟(图7)。这个延迟固定在(0.5秒+1.5 NTSC帧期间+1.5 NTSC帧期间)=0.6秒。

因此，PacketCountDelay是等于0.7秒-0.6秒=0.1秒的常数。同等于这个延迟的Tq刻度数量是：PacketCountDelayIndex =(PacketCountDelay / TqPeriod)。

QLP 130使输入封包计数信息与当前Tq中断同步，如下所示。

针对每个环形缓冲器，QLP维持10位读取指针。首先，QLP搜寻环形缓冲器中的输入项，其tqIndex符合(CurrentTqIndex - PacketCountDelayIndex)值。之后，每隔Tq刻度，QLP以壹为增量递增读取指针值。QLP也检查环形缓冲器中与封包计数一起储存的TqIndex的连续性。如果有中断，则QLP会设定KP 140警示旗标，并且通过搜寻符合(CurrentTqIndex - PacketCountDelayIndex)的TqIndex来初始化读取指针值。

针对所有的输入帧，QLP计算帧期间的平均输入位速率。在计算帧的需求参数时会同时执行这项计算。平均输入位速率用来计算目标帧大小。

首先，QLP计算帧所横跨的整数Tq期间数量。

FrameTqCount=(下一帧译码时间与当前帧译码时间之间的差值)/TqPeriod，调整为下一个较高的整数。

QLP从FrameTqCount计算帧的持续时间：

FrameDuration = FrameTqCount * TqPeriod.

然后，QLP计算平均输入位速率：

AvgInBitrate = InPacketCount * 188 * 8 * / FrameDuration,

其中InPacketCount是从当前读取指针开始，整个视频封包计数环形缓冲器的FrameTqCount输入项的PacketCount总和。

6. 带宽配置

于所有的Tq时隙，QLP执行带宽配置程序。QLP先将带宽指派给不需要经过代码转换处理的纯粹通过视频节目，及数据及音讯节目。

然后，依据频道的需求参数值，将剩余的带宽配置给剩余的频道，并且受限于最大及最小位速率限制条件。

6.1. 通过视频及数据频道

QLP 130将代码转换位速率指派给纯粹通过频道，如下所示：

if(purePassThrough)

TcodeBitrate = VideoInBitrate

其中VideoInBitrate是瞬间输入视频位速率，其计算方式如下：

VideoInBitrate=(于当前读取指针，储存在对应视频节目环形缓冲器输入项中的PacketCount值)* 188 * 8 / TqPeriod。

针对每个统计复用器群组，QLP计算可供动态配置使用的带宽数量，即，扣除纯粹通过频道的带宽及PES校准添加信号(overhead)位之后可用的带宽数量。统计再复用器群组代表于代码转换器100互相竞争带宽的一组频道。在代码转换器100可使用一个或一个以上统计再复用器群组。

AvailableVideoBitrate = TotalOutputBandwidth-(所有频道的NonVideoInBitrate总和)-(所有纯粹通过频道的TcodeBitrate总和)-(非纯粹通过频道数量* PesOverheadBitrate)。

在这个方程式中，TotalOutputBandwidth是可供输入数据流中视频、音讯及数据服务(包括系统信息)使用的总输出传输(有效负载(payload))带宽。这是使用者设定的统计复用器群组参数。

PesOverheadBitrate是PES校准的平均添加信号位速率，这是如下的常数：

PesOverheadBitrate = 1/2 * 184 * 8 * 30 = 22.08Kbps。

计算瞬间非视频位速率(NonVideoInBitrate)的方式类似于VideoInBitrate，如下所示：

NonVideoInBitrate=(于当前读取指针，储存在对应非视频PID环形缓冲器输入项中的PacketCount值)* 188 * 8 / TqPeriod。

6.2. 代码转换位速率配置

针对每个统计复用器群组，QLP依据下列限制条件，在非通过视频频道之间配置AvailableVideoBitrate：

1. 代码转换位速率的总和=GroupBandwidth。由于可供动态配置

使用的带宽是变量，并且受到传输数据流中通过组件(例如，非视频数据)所占用的带宽影响，所以当有一个以上统计复用器被建构来输出传输复用，则群组会陈述为总可用带宽的百分比。

2. 任何单一TPE上所有非纯緛通过视频频道的平均代码转换位速率总和必须小于TPE上可变长度型编码器(380, 440)最大总处理能力的上限。

3. 针对纯緛通过频道，输入位速率应等于输入位速率。可将频道当作纯緛通过频道处理，例如，维持品质。

4. 针对任何视频频道，输出目标帧大小不可大于输入帧大小。这解释成平均代码转换位速率不可超过平均输入位速率的限制条件。

5. 目标帧大小不可大于最大值，也不可小于最小值，这是为了保护视频缓冲器的规定。

以下概述代码转换位速率配置的程序。

6.2.1. 计算最大帧大小的近似值

给定最大目标帧大小，以防止译码器缓冲器下溢的方式如下：

maxFrameSize=(从代码转换帧第一个位进入代码转换器FIFO 445的时间至帧译码时间期间要传输至译码器452的位数量)-(代码转换帧第一个位进入FIFO时，代码转换器FIFO的级别)。

然而，在计算代码转换位速率时，不知道代码转换帧第一个位进入FIFO时的代码转换器FIFO的级别。因此，计算最大代码转换帧大小近似值的方式如下：

maxFrameSizeEstimate = delayBitsMax - FifoLevel - offsetBitsMax。

delayBitsMax值是最后一次读取FIFO级别至帧译码时间要传输至译码器452的位数量，并且通过下列方式计算：

delayBitsMax = TqPeriod * 从 FrameMarker开始的Ndelay期限，传输位速率队列中传输位速率值的总和，其中：

Ndelay=从读取FifoLevel的时间到译码帧的时间所计数的Tq时隙数量。

FifoLevel值是最近代码转换器的输出FIFO级别。

offsetBitsMax值是最后一次读取FIFO级别时间至目标代码转换帧

第一位进入FIFO时间，进入代码转换器FIFO的近似位数量。给定这个近似值的方式为：等待代码转换的帧的最初(无限)目标帧大小。这等于：

offsetBitsMax=最近输出帧的大小+目前正在代码转换的目标帧大小+目前正在等待代码转换的帧之前的帧的目标帧大小总和。

在最大帧大小的近似值中，假设之后代码转换的帧所产生的位数量等合帧目标，并且QLP的输出队列132中的起始帧目标值未到达最大帧大小，也未到达最小帧大小。

6.2.2. 计算最小帧大小的估计值

给定最小目标帧大小，以防止译码器452溢位的方式如下：

MinFrameSize=(从代码转换帧第一个位进入代码转换器FIFO的时间至下一帧译码时间期间要传输至译码器的位数量)-(译码器缓冲器的大小)-(代码转换帧第一个位进入FIFO时，代码转换器FIFO的级别)。

MinFrameSize与MaxFrameSize之间的关系如下所示：

MinFrameSize=MaxFrameSize+(从当前帧译码时间至下一帧译码时间期间要传输至译码器的位数量)-(译码器缓冲器的大小)。

因此，

MinFrameSizeEstimate = MaxFrameSizeEstimate + DeltaBitsMin - DecoderBufferSize

其中DeltaBitsMin=从当前帧译码时间至下一帧译码时间期间要传输至译码器的位数量，其计算方式是计算传输位速率队列中对应项的总和。

在示范性实施中，DecoderBufferSize是MPEG2主描述主级别(MPEG2 Main Profile, Main Level)缓冲器大小，为1.835 Mbits。

6.2.3. 计算保护缓冲器的最大代码转换位速率

必须设定最大代码转换位速率，以避免译码器缓冲器溢位。帧的目标帧大小被计算为按照平均代码转换位速率与平均输入位速率的比例所比例换算的输入帧大小。因此，从最大帧大小来计算最大代码转换位速率的方式如下，假设代码转换位速率维持固定，直到帧时间结束：

MaxTcodeBitrate=((MaxFrameSize / OrigFrameSize * AvgInBitrate

* FrameTqCount-(从帧开始至当前T_q中断的代码转换位速率的总和)*FrameTqIndex)/(FrameTqCount -FrameTqIndex)，

其中OrigFrameSize是输入帧中的位数量，FrameTqCount是帧时间中的Tq时隙数量，而FrameTqIndex是从帧开始($T_{frameStart}$)的Tq时隙数量。

6.2.4. 计算保护缓冲器的最小代码转换位速率

必须设定最小码转换位速率，以避免译码器缓冲器下溢。针对每个视频服务，计算最小代码转换位速率的方式类似于计算最大代码转换位速率的方式：

MinTcodeBitrate=((MinFrameSize / OrigFrameSize * AvgInBitrate * FrameTqCount -(从帧开始至当前Tq中断的代码转换位速率的总和)* FrameTqIndex)/ (FrameTqCount -FrameTqIndex)。

6.2.5. 计算每个TPE可处理的最大累积位速率

在整个窗口(例如，3个帧期间)，任何单一TPE上所有视频服务之间的平均输出位速率会受到TPE中VLE处理能力的限制，例如，在3个帧窗口期间，限制总处理能力不得超过12 Mbits/秒扩展(spread)(与处理器有关的值)的平均值。在任何Tq期间，计算TPE支持的最大位速率的方式如下：

MaxTpeBitrate=(N_{Tq}* V1eThroughput)-(在过去N_{Tq} -1 Tq中断期间，在TPE上的所有视频频道的代码转换位速率的总和)，

其中N_{Tq}=在平均窗口(例如3 NTSC帧时间(100ms))中的Tq时隙数量；V1eThroughput是就平均位速率(例如，12 Mbits/秒)而言，VLE的总处理能力。

6.2.6. 在视频频道之间分配可用的位速率

下列程序适用于每个统计再复用器群组。

1. 在没有最小及最大位速率限制条件情况下，QLP决定理想的带宽配置。

NominalBitrate=AvailableVideoBitrate/统计复用器群组中的视频频道数量

TotalNeed=所有视频频道的NeedParameter总和

if(TotalNeed > 0)

```
    for(所有视频频道)
    {
        NeedBitrate[channel] = AvailableVideoBitrate *
    NeedParameter [channel] / TotalNeed
    }
} else{
    for(所有视频频道)
        NeedBitrate [channel] = NominalBitrate
}
```

2. 将频道的MinTcodeBitrate指派给每个视频频道。如果MinTcodeBitrate的总和超过AvailabeVideoBitrate，则会与MinTcodeBitrate成正例来分配带宽。

```
    TotalMinBitrate=整个统计复用器的MinTcodeBitrate总和
    if(TotalMinBitrate > AvailableVideoBitrate)
    {
        for(所有视频频道)
        {
    TcodeTcodeBitrate [channel] = MinTcodeTcodeBitrate [channel]
* AvailableVideoBitrate / TotalMinBitrate
    }
    AvailableVideoBitrate=0
    处理代码转换位速率配置。
}else{
    for(所有视频频道){
        TcodeBitrate [channel] = MinTcodeBitrate [channel]
        NeedBitrate [channel] = Max(0, NeedBitrate[channel]
        - MinTcodeBitrate [channel])
    }
    AvailableVideoBitrate = AvailableVideoBitrate -
TotalMinBitrate
}
```

3. 然后，QLP尝试满足使用者最小位速率需求。在应用使用者最小位速率之后，QLP通过MaxTcodeBitrate来限制使用者最小

位速率。

for(所有视频频道)

{

if(UserMinBitrate[channel] > MaxTcodeBitrate [c])

minBitrate [channel] = MaxTcodeBitrate [channel]

else

minBitrate [channel] = UserMinBitrate[channel]

}

ExtraMinBitrate=UserMinBitrate大于MinTcodeBitrate的所有频道的(minBitrate - MinTcodeBitrate)总和。

if(ExtraMinBitrate > AvailableVideoBitrate)

{

for(所有视频频道)

{

if(minBitrate[channel] > MinTcodeBitrate[channel])

{

extraBitrate =(minBitrate[channel] - MinTcodeBitrate[channel])* AvailableVideoBitrate / ExtraMinBitrate

TcodeBitrate[channel] = TcodeBitrate[channel] + extraBitrate

needBitrate [channel] = Max(0, needBitrate[channel] - extraBitrate)

}

}

AvailableVideoBitrate=0

}else{

for(所有视频频道)

{

if(minBitrate[channel] > MinTcodeBitrate[channel])

{

extraBitrate = minBitrate[channel] - MinTcodeBitrate[channel]

TcodeBitrate[channel] = TcodeBitrate[channel] + extraBitrate

needBitrate [channel] = Max(0, needBitrate[channel] - extraBitrate)

AvailableVideoBitrate = AvailableVideoBitrate - extraBitrate

}

}

}

4. QLP依据使用者最大位速率、保护译码器缓冲器的最大和最小代码转换位率以及每个TTPE可支持的最大处理位速率，以计算每个频道的最大位速率。

for(所有TPE)

{

tpeAvailableBitrate = MaxTpeBitrate[tpeIndex]- 整个 TPE 的 TcodeBitrate总和

tpeNeedBitrate=整个TPE的NeedBitrate总和

for(TPE处理的所有频道)

MaxBitrate[channel] = Min MaxTcodeBitrate [channel],(tpeAvailableBitrate * NeedBitrate[channel] / tpeNeedBitrate)+ TcodeBitrate [channel], Max (UserMaxBitrate[channel], MinTcodeBitrate [channel]))

}

5. QLP按照剩余NeedBitrate值的比例来指派剩余的带宽

TotalNeedBitrate=所有视频频道的needBitrate总和

for(所有视频频道){

TcodeBitrate[channel] = TcodeBitrate[channel] +(AvailableVideoBitrate * NeedBitrate[channel] / TotalNeedBitrate)

}

AvailableVideoBitrate=0

6. QLP针对位速率配置套用最大位速率限制条件

for(所有视频频道)

{

if(TcodeBitrate[channel] > MaxBitrate[channel])

```
        {
        TcodeBitrate [channel] = MaxBitrate[channel]
        AvailableVideoBitrate = AvailableVideoBitrate +
    TcodeBitrate[channel] - MaxBitrate[channel]
        NeedBitrate [channel]=0
    }
}
```

7. QLP配置从超过最大位速率的频道所收集来的额外带宽

```
TotalNeedBitrate=所有频道的NeedBitrate总和
if(AvailableVideoBitrate>0)
{
    for(所有视频频道)
    {
extraBitrate = AvailableVideoBitrate * NeedBitrate[channel] /
TotalNeedBitrate
if(extraBitrate + TcodeBitrate[channel] > MaxBitrate [channel])
extraBitrate=MaxBitrate[channel]-TcodeBitrate [channel]
TcodeBitrate[channel] = TcodeBitrate[channel] + extraBitrate
AvailableVideoBitrate = AvailableVideoBitrate - extraBitrate
}
}
```

8. QLP按照当前配置的位速率与最大位速率之间差值的比例，配置剩余的带宽。

```
if(AvailableVideoBitrate>0)
{
    TotalHeadroom= 所 有 频 道 的 (MaxBitrate[channel] -
    TcodeBitrate[channel])总和
    for(所有频道)
    {
TcodeBitrate[channel] = TcodeBitrate[channel] +
AvailableVideoBitrate *(MaxBitrate[channel] -
TcodeBitrate[channel] )/ TotalHeadroom
    }
```

}

QLP维护每个视频频道的代码转换位速率队列在每个Tq中断中，计算而得的代码转换位速率值储存在队列中，并且于0.5秒后撷取，以当作传输位速率值使用。

6.2.7. 起始目标帧大小计算

于帧的最后Tq时隙，QLP计算目标帧大小起始值的方式如下。

InitialTargetFrameSize=OrigFrameSize * AvgInBitrate / AvgTcodeBitrate,

其中AvgTcodeBitrate是帧的平均代码转换位速率，定义为帧所占用的所有Tq时隙期间的TcodeBitrate总和。

此时，TPE可能尚未准备好要代码转换新的帧，因此QLP维护每个视频频道的目标帧大小队列。InitialTargetFrameSize值储存在对应频道的队列中，并且当TPE准备要代码转换帧时撷取该值。

6.3. 通过决策

在帧的第一个Tq中断，QLP决定是否通过帧。通过决策依据在新帧开始时，每个频道的帧第一Tq时隙时计算的代码转换位速率，如下所示。

PassThroughBitrate = PassThroughMargin * OrigFrameSize / FrameTqCount;

其中PassThroughMargin是小于但接近1.0的参数，例如0.95；OrigFrameSize是输入帧中的位数量；而FrameTqCount是帧中的Tq时隙数量。使用PassThroughMargin可允许帧大小稍微高于目标帧大小的输入帧通过，藉此维持帧品质，并且也节省代码转换器处理周期。

if(TcodeBitrate > PassThroughBitrate)

{

通过整个帧。

}else{

代码转换帧。

}

6.4. 目标帧大小计算

一旦QLP接收到TPE发出TPE准备好代码转换新帧的信号的讯

息，QLP随即依据最新缓冲器级别信息来计算最大帧大小值及最小帧大小值。然后，QLP从目标帧大小队列132提取目标帧大小，并且使用最大和最小帧大小限制条件来计算目标帧大小的最后值。

6.4.1. 计算最大帧大小

QLP计算最大帧大小，以防止译码器缓冲器下溢。计算方式类似于在Tq中断期间计算最大帧大小近似值的方式(6.2.1)：

MaxFrameSize=DelayBits-FifoLevel-LastOutputFrameSize

。

DelayBits值是从读取FIFO级别时间至帧译码时间期间要传输至译码器的位数量，其计算方式是计算传输位速率队列中目前对应传输位速率值的总和。

FifoLevel值是代码转换器锁定的代码转换器FIFO级别。即，代码转换器会读取FifoLevel并传送至QLP。

6.4.2. 计算最小帧大小

QLP计算最小帧大小，以防止译码器缓冲器下溢。计算方式类似于在Tq中断期间计算最小帧大小近似值的方式(6.2.2)。最小帧大小与最大帧大小之间的关系如下：

MinFrameSize=MaxFrameSize+(从当前帧译码时间至下一帧译码时间期间要传输至译码器的位数量)-(译码器缓冲器的大小)。

如上文所述，就MPEG2主描述主级别(MPEG2 Main Profile Main Level)而言，译码器的缓冲器大小为1.835 Mbits。

6.4.3. 计算前一帧的进位(carryover)

代码转换器可能无法产生完全等于目标帧大小的位数量。出自代码转换前一帧的过剩或不足位会被并入当前帧的目标帧大小。计算这个误差(过剩或不足)的方式如下：

FrameCarryOver=LastOutputFrameSize-(前一帧的TargetFrameSize)。

6.4.4. 计算目标帧大小

QLP从对应视频频道的目标帧大小队列提取InitialTargetFrameSize值，并且通过最大值和最小值来限制目标帧大小：

TargetFrameSize = Min(MaxFrameSize,(Max MinFrameSize,

InitialTargetFrameSize + FrameCarryOver)).

然后，QLP将MinFrameSize值、MaxFrameSize值和TargetFrameSize值传送至TPE。这些值用来引导代码转换处理程序的速率控制。

6.5. 量化控制

在帧内，下列算法用来计算要代码转换的所有宏区块的量化比例值。计算新量化比例Q_{New}的方式是，通过目标位缩减比率R_{New}/R_{Old}，按比例换算输入宏区块Q_{Old}的量化比例。通常，每个宏区块都具有一个量化比例。然而，如切片(slice)或其它编组之类的宏区块群组可与通用量化比例关联。在此情况下，会针对群组决定新的量化比例。

初始化：

R_{Old}=帧中的原始位数量。

R_{New}=TargetFrameSize=代码转换/重新量化帧所产生的目标位数量。

针对所有切片(slice)，请执行下列步骤：

{

$Q_{New}=Q_{Old}*R_{Old}/R_{New}$

/*重新量化切片(slice)后更新R_{Old}和R_{New}：*/

$R_{Old}=R_{Old}$-切片(slice)中的原始位数量。

$R_{New}=R_{New}$-代码转换(例如，包括重新量化)切片(slice)所产生的新位数量。

}

针对Q_{New}，可使用调整成下一个较高的整数，或调整成最接近的整数。前面的公式应导致将帧代码转换成目标帧大小。代码转换器的帧大小可超过目标，但不应超过最大帧大小。

然而，如果帧开始的量化比例较低，藉此产生大量的位，则可能会发生帧中很早就到达最大帧大小，在此情况下，将量化比例设定为最大级别(粗量化)，并且其余帧的品质必然会非常不佳。为了避免此情况，配置每宏区块的最小位数量mb_budget。当代码转换帧时，如果使用的位数量运行计数增加太大(即，使用的位数量到达特定级别，其被调整为帧重新量化的进展)，则会设定短时间使用的应急量

化器，直到为剩余的宏区块留下足够的位可接受mb_budget位数量。也就是说，panic_level尝试MB接受mb_budget或较小位数量的量化比级别。以此方式在整个帧期间扩展应急量化器，使得只有帧的一部份可进入应急模式。为了实现此目的，于每个帧开始时，初始化下列变量：

$$\text{mb\_budget}=\frac{TargetFrameSize}{number\_mbs}\cdot\xi$$

panic_level = MaxFrameSize - mb_budget * number_mbs

ξ使用目标帧大小来决定配置给每个宏区块的最小位数量，作为每宏区块平均位量的分数。范围是0<ξ<1。1/4-1/2的ξ可能适用于大部份的案例。如果ξ太大，则可能会太早触发应急状况；如果ξ是零，则会太晚触发应急状况，使得其余帧陷于应急模式中。

编码每个宏区块之后，

panic_level = panic_level - bits_used_mb + mb_budget

if(panic_level<0)

Q_{New}=MAX_QL;

其中MAX_QL=112(例如，系统可应用的最大QL)。

如果帧大小小于最小帧大小，则会在比特流结尾附加零，使得帧大小等于或大于最小帧大小。

6.6. PCR时隙

MPEG标准要求以100ms最大时间间隔传送PCR (Program Clock Reference；节目时钟参考)。在传输时间之前不知道实际的PCR值，所以代码转换器建立PCR的预留位置(placeholder)时隙。

QLP从目标帧大小估计传输帧的时间，因此会在帧中插入必要的最小PCR数量，以满足最大PCR时间间隔需求。请注意，在满足这个需求过程中，虽然未编码的图片具有固定持续时间(1/30秒)，但是每个帧的已编码比特流可具有可变持续时间。例如，如果帧具有100,000位并且以1 Mbps传输，则持续时间为0.1秒。如果帧以2 Mbps传输，则持续时间为0.05秒。评估传输帧所需的总时间(或者，确切而

言，从代码转换帧第一个位离开代码转换器输输出缓冲器(FIFO)445的时间至帧最后个位离开FIFO的时间所经过的时间)的方式如下：

TxFrameDuration = TargetFrameSize/(传输位速率队列中的最小值)。

于帧期间要插入的最小PCR数量为：

MinPcrCount = TxFrameDuration / MaxPcrSeparation，调整为最接近的整数，

其中MaxPcrSeparation是按照MPEG(100 ms)要求的PCR之间的最大分隔。MaxPcrSeparation=80 ms值用来提供20 ms边缘。

因此，可得知，本发明揭示一种用来处理包含视频数据的数个频道的高效率统计再复用器。在本发明的一项观点中，当从该数据采集统计信息时，会延迟视频数据的代码转换。数据的位传输率需求参数依据该统计信息来决定，并且视频数据的代码转换依据延迟之后的各自位传输率需求参数。

在本发明的另一项观点中，于连续时间间隔的多个时段期间会在统计再复用器更新视频帧的代码转换位传输率，以允许更密切监控位传输率。另外，于每段时间间隔会更新代码转换位传输率的最小及最大界限。因此，在第一时间间隔会代码转换帧的一部份，然后更新代码转换位传输率，接着在第二时间间隔会代码转换帧的第二部份，以此类推。

在本发明还有一项观点中，会按比例换算帧中宏区块的预先代码转换量化度量(quantization scale)，以依据帧中的预先代码转换数据量与该帧的目标后置代码转换数据量的比例，提供对应的新代码转换量化度量。另外，随着代码转换的进展，会针对帧的不同部份来调整量化比例，以确保将最低代码转换带宽量配置给每个宏区块。

虽然已配合各种较佳具体实施例来说明本发明，但是应明白可对本发明进行各种修及调整，而不会脱离本发明权利要求所提出的本发明的范畴。

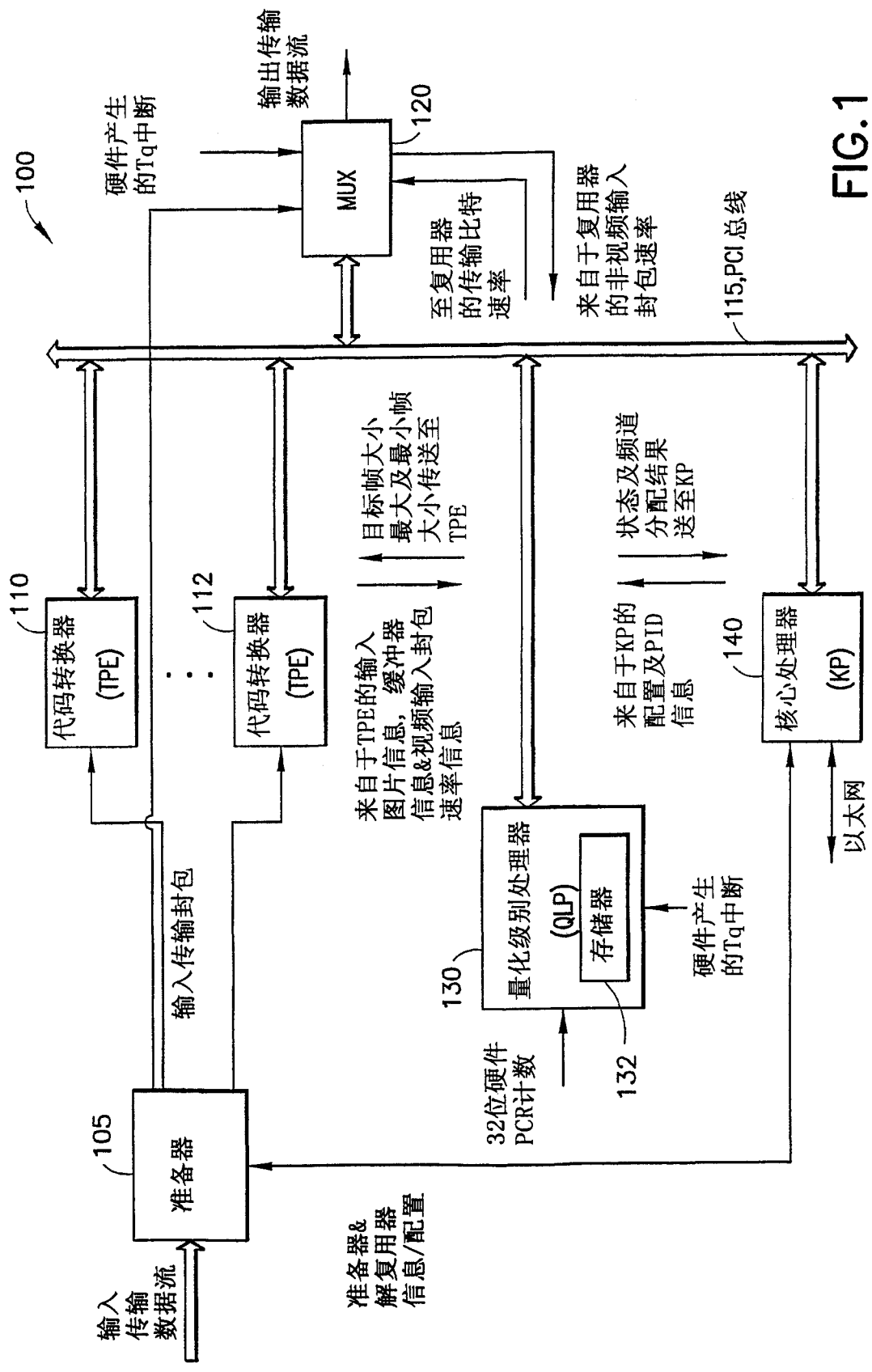

100
硬件产生的Tq中断
输出传输数据流
MUX
120
至复用器的传输比特速率
来自于复用器的非视频输入封包速率
115,PCI 总线
110
代码转换器 (TPE)
112
代码转换器 (TPE)
目标帧大小 最大及最小帧大小传送至TPE
来自于TPE的输入图片信息，缓冲器信息&视频输入封包速率信息
状态及频道分配结果送至KP
来自于KP的配置及PID信息
140
核心处理器 (KP)
以太网
输入传输封包
量化级别处理器 (QLP)
存储器
130
132
32位硬件PCR计数
硬件产生的Tq中断
105
准备器
准备器&解复用器信息/配置
输入传输数据流

FIG. 1

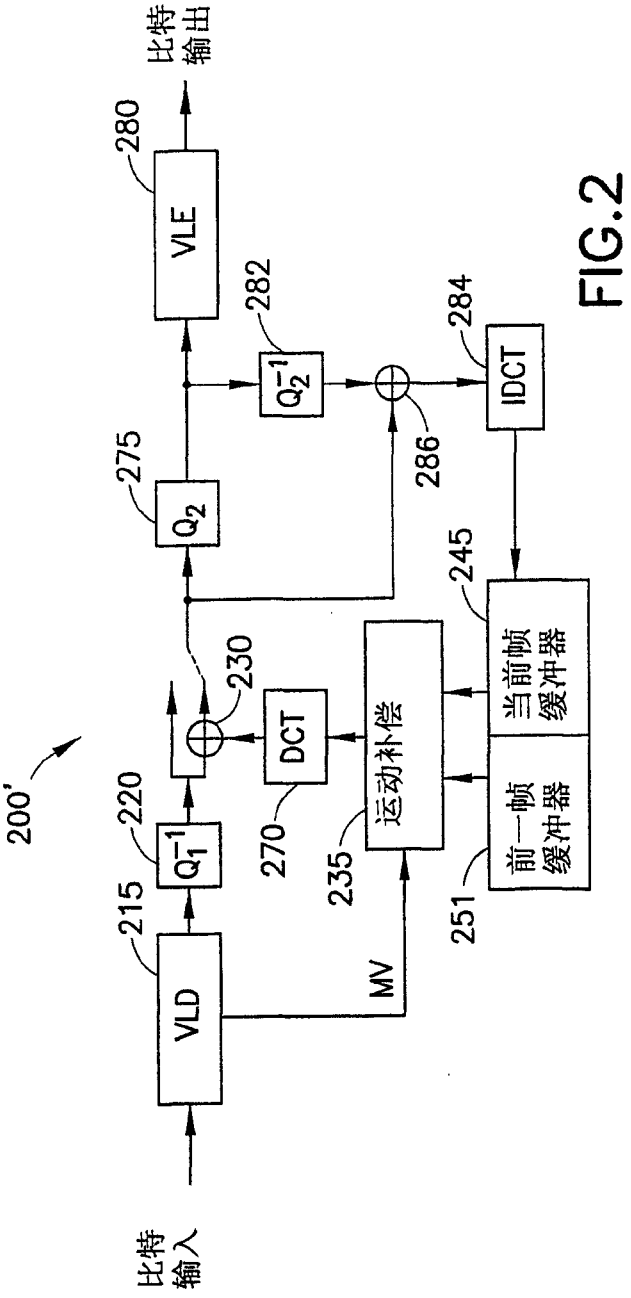

200'
比特输入
VLD
215
Q_1^{-1}
220
230
DCT
270
运动补偿
235
MV
前一帧缓冲器
251
当前帧缓冲器
245
Q_2
275
VLE
280
比特输出
Q_2^{-1}
282
IDCT
284
286

FIG.2

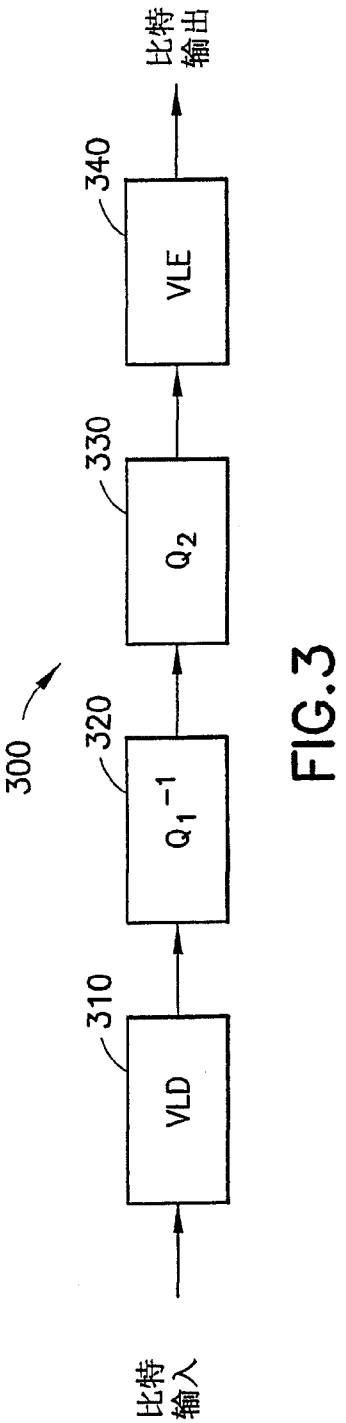

300
比特输入
VLD
310
Q_1^{-1}
320
Q_2
330
VLE
340
比特输出

FIG.3

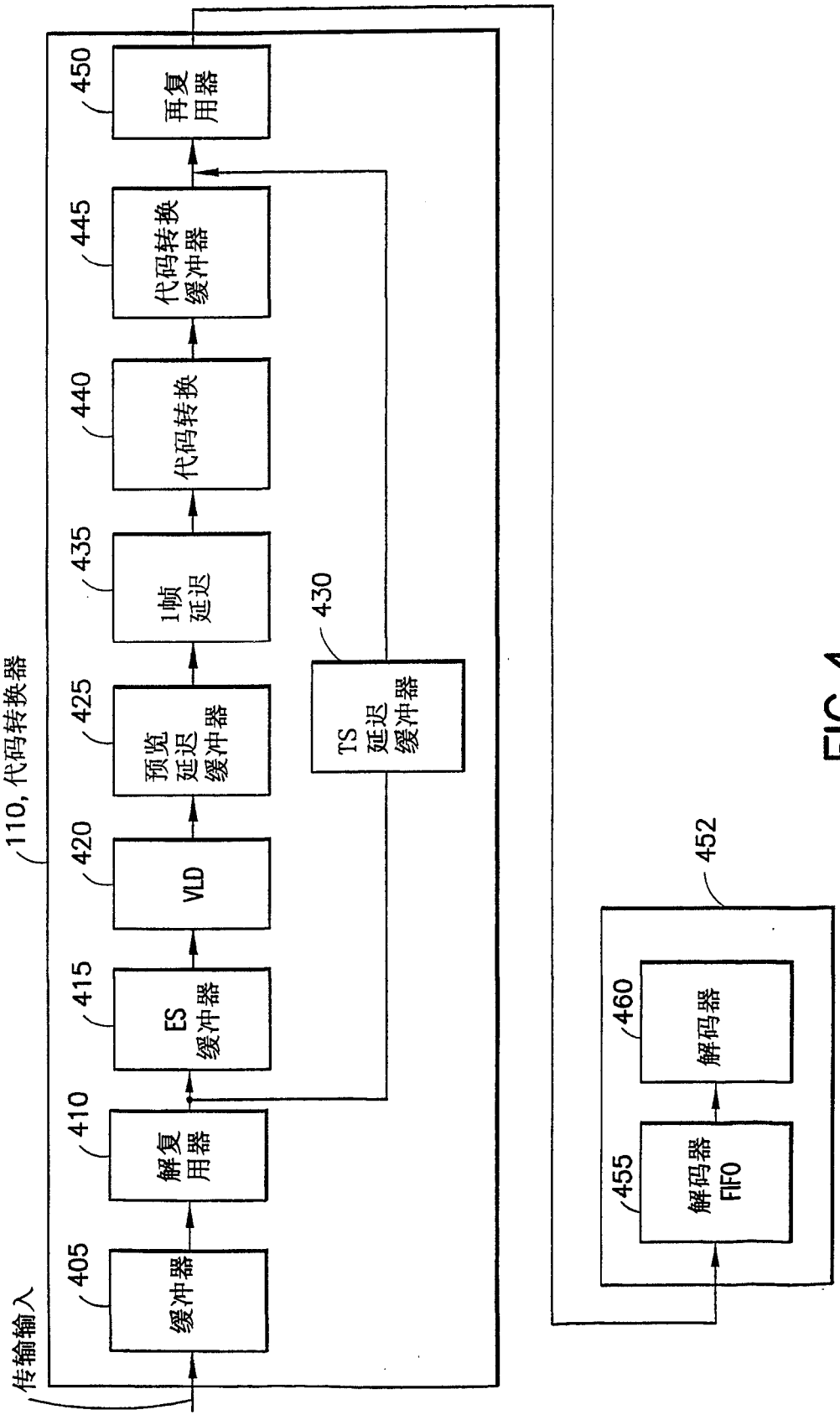

传输输入
110, 代码转换器
405
缓冲器
410
解复用器
415
ES缓冲器
420
VLD
425
预览延迟缓冲器
430
TS延迟缓冲器
435
1帧延迟
440
代码转换
445
代码转换缓冲器
450
再复用器
452
455
解码器FIFO
460
解码器

FIG.4

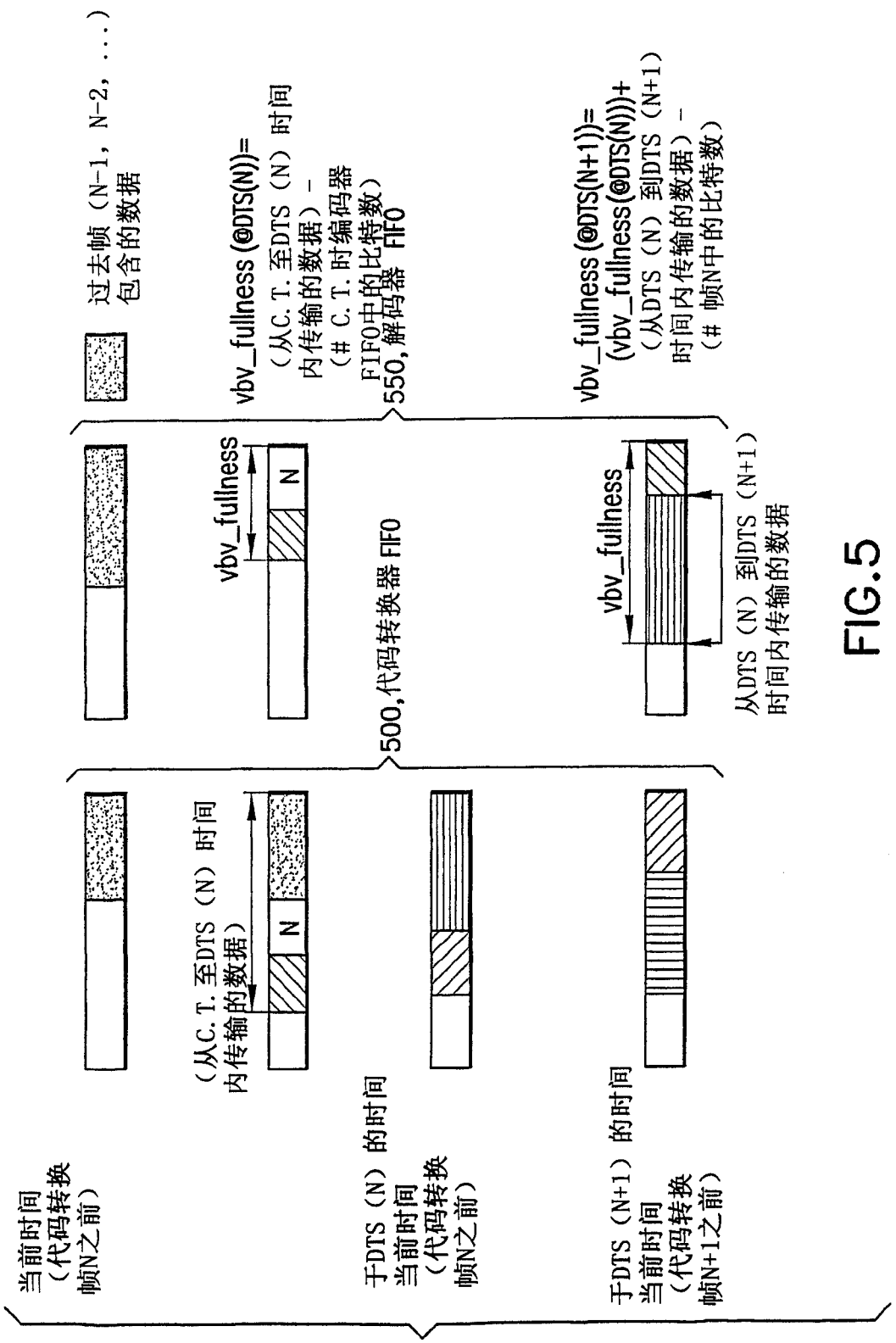

当前时间
（代码转换
帧N之前）
（从C. T. 至DTS（N）时间
内传输的数据）
N
于DTS（N）的时间
当前时间
（代码转换
帧N之前）
于DTS（N+1）的时间
当前时间
（代码转换
帧N+1之前）
500,代码转换器 FIFO
vbv_fullness
N
vbv_fullness
从DTS（N）到DTS（N+1）
时间内传输的数据
过去帧（N-1，N-2，...）
包含的数据
vbv_fullness (@DTS(N))=
（从C. T. 至DTS（N）时间
内传输的数据）-
（# C. T. 时编码器
FIFO中的比特数）
550,解码器 FIFO
vbv_fullness (@DTS(N+1))=
(vbv_fullness(@DTS(N)))+
（从DTS（N）到DTS（N+1）
时间内传输的数据）-
（# 帧N中的比特数）

FIG.5

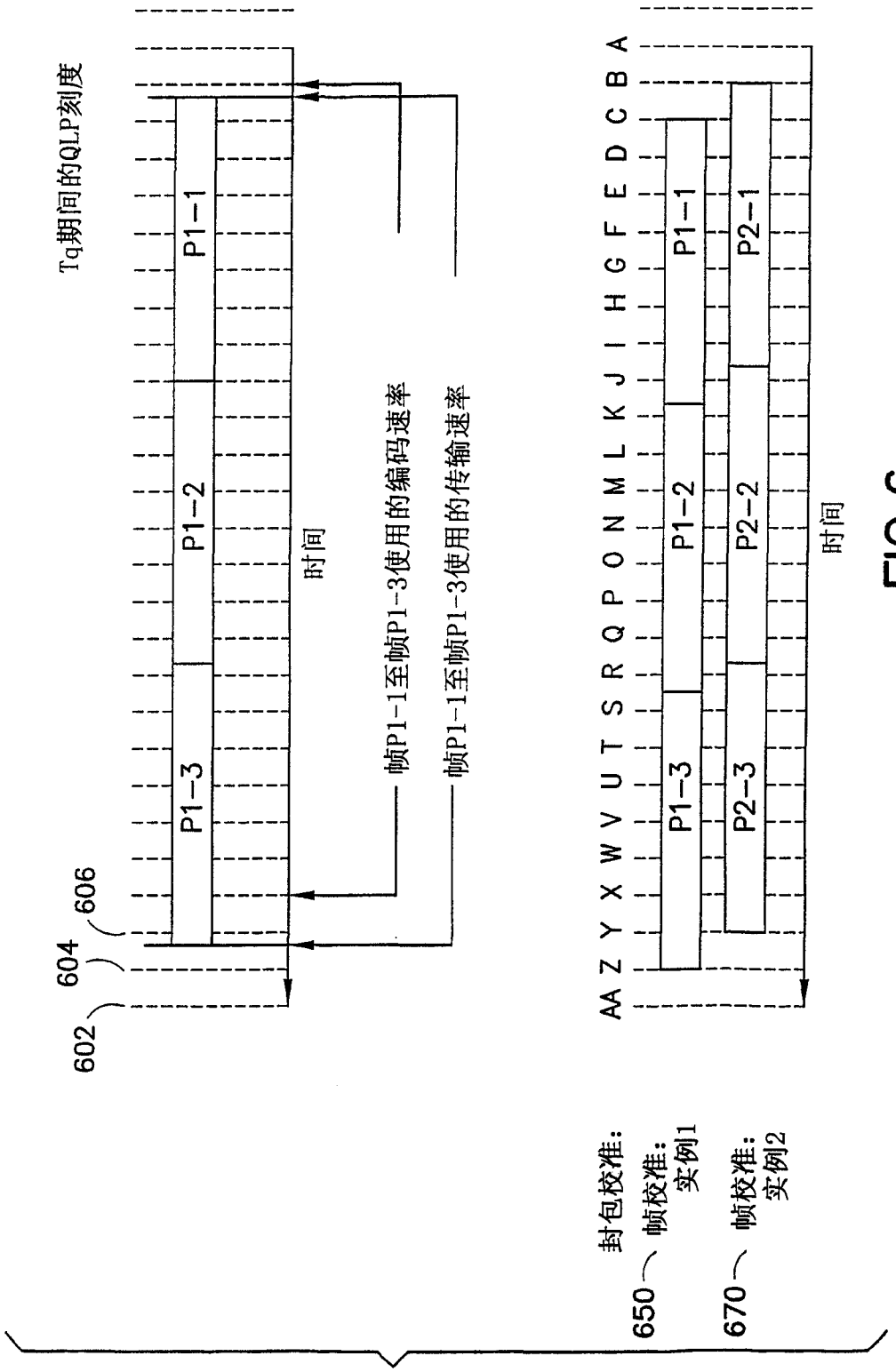

Tq期间的QLP刻度
602
604
606
P1-3
P1-2
P1-1
时间
帧P1-1至帧P1-3使用的编码速率
帧P1-1至帧P1-3使用的传输速率
AA Z Y X W V U T S R Q P O N M L K J I H G F E D C B A
P1-3
P1-2
P1-1
P2-3
P2-2
P2-1
时间
封包校准:
650
帧校准:
实例1
670
帧校准:
实例2

FIG.6

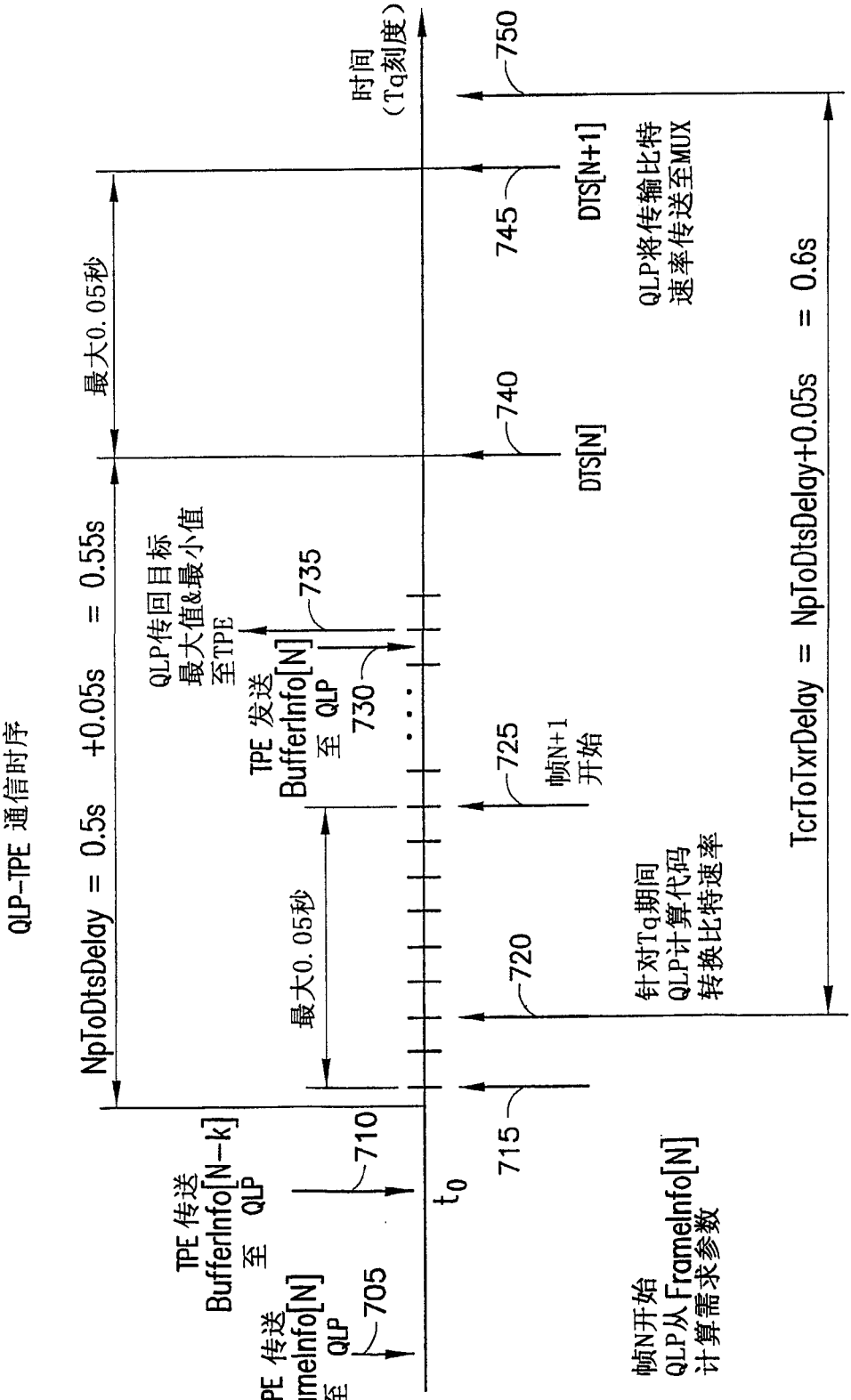

QLP-TPE 通信时序
NpToDtsDelay = 0.5s +0.05s = 0.55s
最大0.05秒
时间
(Tq刻度)
TPE 传送
BufferInfo[N-k]
至 QLP
TPE 传送
FrameInfo[N]
至 QLP
705
710
t0
715
最大0.05秒
TPE 发送
BufferInfo[N]
至 QLP
730
735
QLP传回目标
最大值&最小值
至TPE
720
帧N开始
QLP从FrameInfo[N]
计算需求参数
针对Tq期间
QLP计算代码
转换比特速率
725
帧N+1
开始
740
DTS[N]
745
DTS[N+1]
750
QLP将传输比特
速率传送至MUX
TcrToTxrDelay = NpToDtsDelay+0.05s = 0.6s

FIG.7