一种数据处理方法和数据处理装置

本申请要求于2022年01月28日提交中国专利局、申请号为202210109956.X、发明名称为“一种数据处理方法和数据处理装置”的中国专利申请的优先权、于2022年03月23日提交中国专利局、申请号为202210290887.7、发明名称为“一种数据处理方法和数据处理装置”的中国专利申请的优先权、于2022年09月01日提交中国专利局、申请号为202211065772.4、发明名称为“一种数据处理方法和数据处理装置”的中国专利申请的优先权、于2022年10月24日提交中国专利局、申请号为202211305113.3、发明名称为“一种数据处理方法和数据处理装置”的中国专利申请的优先权和于2022年11月18日提交中国专利局、申请号为202211448533.7、发明名称为“一种数据处理方法和数据处理装置”的中国专利申请的优先权，其全部内容通过应用结合在本申请中。

技术领域

本申请涉及通信领域，尤其涉及一种数据处理方法和数据处理装置。

背景技术

在5G、云计算、大数据和人工智能等持续推动下，光通信系统及光传输网（optical transport network，OTN）正朝着大容量和超高速方向发展。采用前向纠错编码（forward error correction，FEC）对传输的数据进行纠错，能够解决传输误码，从接收数据中恢复出发送端发送的原始数据。

当前提出有一种级联FEC的传输方案，发端设备和发端处理模块通过连接单元接口（attachment unit interface，AUI）连接。发端设备对待传输数据进行第一FEC编码，并将第一FEC编码后的数据发送至发端处理模块。发端处理模块对第一FEC编码后的数据再进行第二FEC编码，并将第二FEC编码后的数据通过信道传输至数据接收端。具体地，发端处理模块会收到多路数据流，要分别先对多路数据流先进行卷积交织，再对卷积交织后的每一路数据流进行第二FEC编码。应理解，为了提升性能应当使得参与第二FEC编码的一路数据流来自于第一FEC编码后的多个码字，不过这需要采用时延较长的卷积交织来实现，在要求低时延的场景下应用效果不理想。

发明内容

本申请实施例提供了一种数据处理方法及数据处理装置。可以在采用较低时延的场景下获得较好的级联FEC方案性能。

第一方面，本申请提供了一种数据处理方法，该方法包括如下步骤。首先，对n条通道数据流分别进行卷积交织得到n条第一数据流。其中，n为大于1的整数，n条通道数据流都经过第一FEC编码。第一FEC编码后的每a个码字分布在b条通道数据流中，a≤b≤n，n能被b整除，a为大于或等于1的整数。每条第一数据流中连续的z个符号来自z个不同的码

字，z 为大于 1 的整数。之后，对 n 条第一数据流中的每 K 条第一数据流进行复用得到 1 条第二数据流，以得到共 m 条第二数据流。其中，n 条第一数据流包括 G 个第一数据流子集，不同第一数据流子集中的符号来自不同码字，m=n/K，K 为大于 1 的整数，G 为大于 1 的整数。每条第二数据流中连续的 y 个符号来自 y 个不同的码字，y＞z。若 K≤G，则 K 条第一数据流分别来自 K 个第一数据流子集。若 K＞G，则 K 条第一数据流包括每个第一数据流子集中的 K/G 条第一数据流。

在该实施方式中，n 路通道数据流都为经过外码编码后的码字流，在对 n 路数据流分别进行卷积交织处理，并对卷积交织后的 n 路数据流进行数据流复用为 m 路第二数据流，随后进行内码编码。采用本申请提供的数据交织和复用处理方案，通过较短的时延就能实现复用处理后的 m 路数据流在连续输出的多个符号来自多个不同的外码码字，使得级联 FEC 方案在保证较好性能的基础上也有助于降低数据交织的时延。也就是说，本申请中卷积交织处理与数据复用相结合的方案使得级联 FEC 方案的整体时延较低，更适用于要求低时延的应用场景。

在一些可能的实施方式中，对 1 条通道数据流进行卷积交织得到 1 条第一数据流包括：根据 p 条延迟线对 1 条通道数据流进行延迟得到 1 条第一数据流。其中，p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同。存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 d 个符号，z=p*d。每条通道数据流中的符号按照 p 条延迟线的序号依次输入 p 条延迟线，每条延迟线单次输入 d 个符号且单次输出 d 个符号，1 条第一数据流中连续的 p*d 个符号包括每条延迟线输出的 d 个符号。Q 为大于或等于 1 的整数，d 为大于或等于 1 的整数。在该实施方式中，提供了一种卷积交织的具体实现方式，提高了本方案的实用性。

在一些可能的实施方式中，p 条延迟线中序号最大的延迟线包括 0 个存储单元。其中，$d(p*Q+1) \geq a*N/b$，d≤a， N 为码字的长度，从而可以实现每条第一数据流中连续的 z 个符号来自 z 个不同的码字。

在一些可能的实施方式中，p 条延迟线中序号最小的延迟线包括 0 个存储单元。其中，$d(p*Q-1) \geq a*N/b$，d≤a，N 为码字的长度，从而可以实现每条第一数据流中连续的 z 个符号来自 z 个不同的码字。

在一些可能的实施方式中，若 K≤G，则 y=K*z；若 K＞G，则 y=G*z。采用这种复用方式可以保证在多个不同应用场景中都能实现 y＞z，通过较低的时延就能实现较好的级联 FEC 方案的性能。

在一些可能的实施方式中，每条第二数据流包括多个第二数据流符号子集，每个第二数据流符号子集包括 K 组符号，每组符号包括Δ个符号。其中，每个第二数据流符号子集中相邻两组符号来自不同的第一数据流子集。若 K≤G，则Δ为 z 的约数；若 K>G,则Δ= z。

在该实施方式中，由于每个第二数据流符号子集中相邻两组符号来自不同的第一数据流子集，所以经过复用后的第二数据流中连续的 y 个符号来自不同的 y 个码字，其中，y＞z（y=K*z 或 y=G*z）。应理解，如果只通过卷积交织想得到输出数据流中连续的 y 个符号来自不同的 y 个码字，所需要的延时较高。而本方案中，在减少了卷积交织所占用时长的基础上，结合复用处理仍能达到相当的性能。并且，进行复用处理所占用的时长较短，结合卷积交织和复用

就可以通过更短的延时实现相当的性能。

在一些可能的实施方式中，每个第二数据流符号子集中第 j 组符号来自参与复用的 K 条第一数据流中第 j 条第一数据流，0≤j≤K-1。通过上述方式，提供了一种对参与复用的 K 条第一数据流进行选取的规则，保证了每个第二数据流符号子集中相邻两组符号来自不同的第一数据流子集。

在一些可能的实施方式中，若 K＞G，参与复用的 K 条第一数据流中相邻的 2 条第一数据流来自不同的第一数据流子集。通过上述方式，在 K＞G 的场景下，提供了一种对参与复用的 K 条第一数据流进行选取的规则，保证了 y=G*z。

在一些可能的实施方式中，若 K＞G，参与复用的每连续 G 条第一数据流分别来自不同的第一数据流子集。通过上述方式，在 K＞G 的场景下，提供了一种对参与复用的 K 条第一数据流进行选取的规则，进一步保证了 y=G*z。

在一些可能的实施方式中，n=32，K=2、4 或 8。通过上述方式，提供了复用器的几种具体类型，扩展了本方案的应用场景。

在一些可能的实施方式中，n=32，p=2、3、4、6 或 8，d=1 或 2。通过上述方式，提供了卷积交织器的几种具体类型，扩展了本方案的应用场景。

在一些可能的实施方式中，a=1 或 2，b=4、8、16。通过上述方式，提供了通道数据流的几种分布方式，扩展了本方案的应用场景。

在一些可能的实施方式中，对 n 条通道数据流分别进行卷积交织得到 n 条第一数据流之前，方法还包括：对 n 条通道数据流进行通道重排序，以使得 n 条通道数据流按照预设顺序排列。

在一些可能的实施方式中，对 n 条通道数据流分别进行卷积交织得到 n 条第一数据流之前，方法还包括：对 n 条通道数据流进行通道纠偏。在该实施方式中，提供了一种通道数据对齐的具体实现方式，增强了本方案的可实现性。

在一些可能的实施方式中，对 n 条通道数据流分别进行卷积交织得到 n 条第一数据流之前，方法还包括：对 n 条通道数据流进行对齐，以使得 n 条通道数据流中的符号对齐。在该实施方式中，提供了另一种通道数据对齐的具体实现方式，增强了本方案的灵活性。

在一些可能的实施方式中，得到共 m 条第二数据流之后，方法还包括：对 m 条第二数据流分别进行第二 FEC 编码。第二 FEC 编码的信息位长度小于或等于 y 个符号。

第二方面，本申请提供了一种数据处理装置。该数据处理装置包括卷积交织器和复用器。卷积交织器用于：对 n 条通道数据流分别进行卷积交织得到 n 条第一数据流。其中，n 为大于 1 的整数，n 条通道数据流都经过第一 FEC 编码。第一 FEC 编码后的每 a 个码字分布在 b 条通道数据流中，a≤b≤n，n 能被 b 整除，a 为大于或等于 1 的整数。每条第一数据流中连续的 z 个符号来自 z 个不同的码字，z 为大于 1 的整数；复用器用于：对 n 条第一数据流中的每 K 条第一数据流进行复用得到 1 条第二数据流，以得到共 m 条第二数据流。其中，n 条第一数据流包括 G 个第一数据流子集，不同第一数据流子集中的符号来自不同码字，m=n/K，K 为大于 1 的整数，G 为大于 1 的整数。每条第二数据流中连续的 y 个符号来自 y 个不同的码

字，y＞z。若K≤G，则K条第一数据流分别来自K个第一数据流子集。若K＞G，则K条第一数据流包括每个第一数据流子集中的K/G条第一数据流。

在一些可能的实施方式中，卷积交织器具体用于：根据p条延迟线对1条通道数据流进行延迟得到1条第一数据流。其中，p为大于1的整数，每条延迟线包括的存储单元数量各不相同。存储单元数量最小的延迟线包括0个存储单元，每相邻两条延迟线的存储单元数量的差值为Q，每个存储单元用于存储d个符号，z=p*d。每条通道数据流中的符号按照p条延迟线的序号依次输入p条延迟线，每条延迟线单次输入d个符号且单次输出d个符号，1条第一数据流中连续的p*d个符号包括每条延迟线输出的d个符号,Q为大于或等于1的整数，d为大于或等于1的整数。

在一些可能的实施方式中，p条延迟线中序号最大的延迟线包括0个存储单元，其中，$d(p*Q+1)\geq a*N/b$，N为码字的长度，d≤a。

在一些可能的实施方式中，p条延迟线中序号最小的延迟线包括0个存储单元，其中，$d(p*Q-1)\geq a*N/b$，N为码字的长度，d≤a。

在一些可能的实施方式中，若K≤G，则y=K*z；若K＞G，则y=G*z。

在一些可能的实施方式中，每条第二数据流包括多个第二数据流符号子集，每个第二数据流符号子集包括K组符号，每组符号包括Δ个符号。其中，每个第二数据流符号子集中相邻两组符号来自不同的第一数据流子集。若K≤G，则Δ为z的约数；若K>G,则Δ= z。

在一些可能的实施方式中，每个第二数据流符号子集中第j组符号来自参与复用的K条第一数据流中第j条第一数据流，0≤j≤K-1。

在一些可能的实施方式中，若K＞G，参与复用的K条第一数据流中相邻的2条第一数据流来自不同的第一数据流子集。

在一些可能的实施方式中，若K＞G，参与复用的每连续G条第一数据流分别来自不同的第一数据流子集。

在一些可能的实施方式中，n=32，K=2、4或8。

在一些可能的实施方式中，n=32，p=2、3、4、6或8，d=1或2。

在一些可能的实施方式中，a=1或2，b=4、8、16。

在一些可能的实施方式中，数据处理装置还包括通道重排序单元，对n条通道数据流分别进行卷积交织得到n条第一数据流之前，通道重排序单元用于：对n条通道数据流进行通道重排序，以使得n条通道数据流按照预设顺序排列。

在一些可能的实施方式中，数据处理装置还包括通道数据对齐单元，对n条通道数据流分别进行卷积交织得到n条第一数据流之前，通道数据对齐单元用于：对n条通道数据流进行通道纠偏。

在一些可能的实施方式中，数据处理装置还包括通道数据对齐单元，对n条通道数据流分别进行卷积交织得到n条第一数据流之前，通道数据对齐单元用于：对n条通道数据流进行对齐，以使得n条通道数据流中的符号对齐。

在一些可能的实施方式中，数据处理装置还包括编码器，得到共m条第二数据流之后，编码器用于：对m条第二数据流分别进行第二FEC编码，第二FEC编码的信息位长度小于或

等于 y 个符号。

第三方面，本申请提供了一种数据处理方法，该方法包括如下步骤。对 n 条通道数据流进行交织处理得到 m 条目标数据流，n 为 4 的倍数。n 条通道数据流都经过第一前向纠错 FEC 编码，第一 FEC 编码后的每 a 个码字分布在 b 条通道数据流中，a≤b≤n，n 能被 b 整除。每条目标数据流中连续的 F 个符号来自 F 个不同的码字，F＞a。每条目标数据流中连续的 F 个符号至少来自 K1 条不同的通道数据流，每条目标数据流中连续的 F 个符号来自 n 条通道数据流对齐的 n 个符号中最多 K2 个符号，K1 和 K2 为 n 的约数，K2 为 K1 的约数。每条目标数据流中连续的 F 个符号中最多有 K3 个符号来自相同的通道数据流，K3=⌈F/K1⌉，⌈F/K1⌉表示对 F/K1 的商进行向上取整的整数，K3 个符号中任意两个符号在同一通道数据流上相隔至少 K4 个符号，K4≥a*N*K2/n，其中 N 为码字长度。

在一些可能的实施方式中，K1=n/4，K2=n/16。

在一些可能的实施方式中，对 n 条通道数据流进行交织处理得到 m 条目标数据流包括：对 n 条通道数据流分别进行卷积交织得到 n 条第一数据流。每条第一数据流中连续的 z 个符号来自至少 c 个不同的码字，z 为大于 1 的整数，a≤c≤F，c*k2≥F。每条第一数据流中连续的 z 个符号中最多 k1/k2 个符号来自同一个码字。对 n 条第一数据流中的每 K1 条第一数据流进行分组交织得到 S 条目标数据流，以得到共 m 条目标数据流。其中，S 为大于或等于 1 的整数，m=S*n/K1，S≥k1/k2,n 条第一数据流包括 K1 个第一数据流分组，同一第一数据流分组中每两条第一数据流的符号来自相同的码字，K1 条第一数据流分别来自 K1 个第一数据流分组。

在一些可能的实施方式中，对 1 条通道数据流进行卷积交织得到 1 条第一数据流包括：根据 p 条延迟线对 1 条通道数据流进行延迟得到 1 条第一数据流。p 为大于 1 的整数,p*a≥F/k2，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元,每相邻两条延迟线的存储单元数量的差值为 Q,每个存储单元用于存储 d 个符号,z=p*d。每条通道数据流中的符号按照 p 条延迟线的序号依次输入 p 条延迟线，每条延迟线单次输入 d 个符号且单次输出 d 个符号，1 条第一数据流中连续的 p*d 个符号包括每条延迟线输出的 d 个符号，Q 为大于或等于 1 的整数，d 为大于或等于 1 的整数，d≤a。

在一些可能的实施方式中，p 条延迟线中序号最大的延迟线包括 0 个存储单元，其中，$d(p*Q+1)\geq K4$。

在一些可能的实施方式中，p 条延迟线中序号最小的延迟线包括 0 个存储单元，其中，$d(p*Q-1)\geq K4$。

在一些可能的实施方式中，参与分组交织的 K 条第一数据流包括第一符号矩阵，第一符号矩阵包括 K 行 B 列个符号，B=R*p*d，R 为大于或等于 1 的整数，经过分组交织后的 S 条目标数据流包括第二符号矩阵，第二符号矩阵包括 S 行 F 列个符号，K*B=S*F。第一符号矩阵中的符号至少来自 F 个不同码字，且第一符号矩阵中最多有 R*K1/K2 个符号来自相同的码字。

在一些可能的实施方式中，第二符号矩阵中每行的 F 个符号来自第一符号矩阵中至少⌈F/K2⌉列，且⌈F/K2⌉列中每列最多取K2个符号，⌈F/K2⌉表示对 F/K2的商进行向上取整的整数。

第二符号矩阵中每行的 F 个符号包括第一符号矩阵中每行的至少⌊F/K1⌋个符号，⌊F/K1⌋表示对 F/K1 的商进行向下取整的整数，且第二符号矩阵中每行的 F 个符号包括第一符号矩阵中每行的最多⌈F/K1⌉个符号，⌈F/K1⌉表示对 F/K 的商进行向上取整的整数。

在一些可能的实施方式中，第二符号矩阵的每一行中来自第一符号矩阵奇数列的符号分别位于第一符号矩阵的不同行。第二符号矩阵的每一行中来自第一符号矩阵偶数列的符号位于第一符号矩阵的不同行。

在一些可能的实施方式中，第二符号矩阵的每一行中由延迟取值相同的延迟线输出的符号来自第一符号矩阵的不同行。

在一些可能的实施方式中，第二符号矩阵的每一行的最多 K3 个符号来自第一符号矩阵的同一行，K3 个符号中任意 2 个符号分别由延迟差值大于或等于 2*Q*d 的 2 条延迟线输出。

在一些可能的实施方式中，对 n 条通道数据流进行交织处理得到 m 条目标数据流包括：对 n 条通道数据流进行第一分组交织得到 T 条第一数据流，每条第一数据流中连续的 C 个符号至少来自 E 个不同的码字，T=n/K1，C 为 a 的倍数，E≥K2*a。对 T 条第一数据流进行卷积交织得到 T 条第二数据流，每条第二数据流中连续的 H 个符号来自至少 F 个不同码字，F≥E，每条第二数据流中连续的 H 个符号最多 K1/K2 个符号来自相同码字。对 T 条第二数据流中每条第二数据流进行第二分组交织得到 S 条目标数据流，以得到共 m 条目标数据流，m=T*S，S≥k1/K2。

在一些可能的实施方式中，参与第一分组交织的 n 条通道数据流包括第三符号矩阵，第三符号矩阵包括 n 行 A 列个符号，A 为 a 的倍数，经过第一分组交织后的 T 条第一数据流包括第四符号矩阵，第四符号矩阵包括 T 行 C 列个符号，T 为 n 的约数，n*A=T*C。第三符号矩阵的一列中每连续 T 个符号为符号子矩阵，第四符号矩阵中每一列的 T 个符号分别与第三符号矩阵中每个符号子矩阵一一对应。

在一些可能的实施方式中，第三符号矩阵中的符号子矩阵按第一顺序排列，第三符号矩阵中每列的第 1 行到第 n 行包括按第一顺序排列的第 1 个符号子矩阵到第 n/T 个符号子矩阵，第三符号矩阵的相邻两列中前一列的第 n/T 个符号子矩阵到后一列的第 1 个符号子矩阵为按第一顺序排列的连续 2 个符号子矩阵，第四符号矩阵中第 1 列的 T 个符号来自第三符号矩阵中按第一顺序排列的第一个符号子矩阵，依次类推，直到第四符号矩阵中第 C 列的 T 个符号来自第三符号矩阵中按第一顺序排列的最后一个符号子矩阵。或者，第三符号矩阵中的符号子矩阵按第二顺序排列，第三符号矩阵中每 T 行的第 1 列到第 A 行包括按第二顺序排列的第 1 个符号子矩阵到第 A 个符号子矩阵，第三符号矩阵的相邻两个连续 T 行中前 T 行的第 A 个符号子矩阵到后 T 行的第 1 个符号子矩阵为按第二顺序排列的连续 2 个符号子矩阵，第四符号矩阵中第 1 列的 T 个符号来自第三符号矩阵中按第二顺序排列的第一个符号子矩阵，依次类推，直到第四符号矩阵中第 C 列的 T 个符号来自第三符号矩阵中按第二顺序排列的最后一个符号子矩阵。

在一些可能的实施方式中，对 1 条第一数据流进行卷积交织得到 1 条第二数据流包括：根据 p 条延迟线对 1 条第一数据流进行延迟得到 1 条第二数据流，p 为大于 1 的整数，p*E≥F，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，

每相邻两条延迟线的存储单元数量的差值为Q，每个存储单元用于存储C个符号，p*C=H，每条第一数据流中的符号按照p条延迟线的序号依次输入p条延迟线，每条延迟线单次输入C个符号且单次输出C个符号，1条第二数据流中连续的p*C个符号包括每条延迟线输出的C个符号，Q为大于或等于1的整数。

在一些可能的实施方式中，p条延迟线中序号最大的延迟线包括0个存储单元，其中，$C(p*Q+1) \geq K1*K4$。

在一些可能的实施方式中，p条延迟线中序号最小的延迟线包括0个存储单元，其中，$C(p*Q-1) \geq K1*K4$。

在一些可能的实施方式中，每条第二数据流包括R个符号集合，每个符号集合包括p个符号子集，每个符号子集包括C个符号，p个符号子集分别由p条延迟线输出，每个符号集合中的符号来自至少F个不同码字，每条目标数据流包括F个符号，R*p*C=S*F，R为大于或等于的整数。目标数据流中F个符号来自至少$\lfloor F/K2*a) \rfloor$个不同的符号子集，且$\lceil F/(K2*a) \rceil$个不同的符号子集中每个符号子集最多取$K2*a$个符号，$\lceil F/(K2*a) \rceil$表示对$F/(K2*a)$的商进行向上取整的整数。

在一些可能的实施方式中，目标数据流中F个符号包括来自第一符号子集中的第一组符号和来自第二符号子集的第二组符号，第一符号子集和第二符号子集属于同一个符号集合，第一符号子集和第二符号子集分别由两条相邻的延迟线输出，第一符号子集中的符号和第二符号子集中的符号分别按次序排列，第一组符号在第一符号子集中的次序与第二组符号在第二符号子集中的次序不同。目标数据流中F个符号包括来自第三符号子集中的第三组符号和来自第四符号子集的第四组符号，第三符号子集和第四符号子集属于不同的符号集合，第三符号子集和第四符号子集由同一条延迟线输出，第三符号子集中的符号和第四符号子集中的符号分别按次序排列，第三组符号在第三符号子集中的次序与第四组符号在第四符号子集中的次序不同。

在一些可能的实施方式中，每条目标数据流中F个符号最多有$\lfloor F/R \rfloor$个符号来自相同的符号集合，$\lfloor F/R \rfloor$表示对F/R的商进行向上取整的整数。

在一些可能的实施方式中，得到共m条目标数据流之后，方法还包括：对m条目标数据流分别进行第二FEC编码，第二FEC编码的信息位长度等于F个符号。

第四方面，本申请提供了一种数据处理装置，该数据处理装置包括交织模块。交织模块用于：对n条通道数据流进行交织处理得到m条目标数据流，n为4的倍数，n条通道数据流都经过第一前向纠错FEC编码，第一FEC编码后的每a个码字分布在b条通道数据流中，a≤b≤n，n能被b整除。每条目标数据流中连续的F个符号来自F个不同的码字，F>a。每条目标数据流中连续的F个符号至少来自K1条不同的通道数据流，每条目标数据流中连续的F个符号来自n条通道数据流对齐的n个符号中最多K2个符号，K1和K2为n的约数，K2为K1的约数。每条目标数据流中连续的F个符号中最多有K3个符号来自相同的通道数据流，K3=$\lceil F/K1 \rceil$，$\lfloor F/K1 \rfloor$表示对F/K1的商进行向上取整的整数，K3个符号中任意两个符号在同一通道数据流上相隔至少K4个符号，K4≥a*N*K2/n，其中N为码字长度。

在一些可能的实施方式中，K1=n/4，K2=n/16。

在一些可能的实施方式中，交织模块包括卷积交织器和分组交织器。卷积交织器用于：对 n 条通道数据流分别进行卷积交织得到 n 条第一数据流，每条第一数据流中连续的 z 个符号来自至少 e 个不同的码字，z 为大于 1 的整数，a≤e≤F，e*k2≥F，每条第一数据流中连续的 z 个符号中最多 k1/k2 个符号来自同一个码字。分组交织器用于：对 n 条第一数据流中的每 K1 条第一数据流进行分组交织得到 S 条目标数据流，以得到共 m 条目标数据流，其中，S 为大于或等于 1 的整数，m=S*n/K1，S≥k1/k2,n 条第一数据流包括 K1 个第一数据流分组，同一第一数据流分组中每两条第一数据流的符号来自相同的码字，K1 条第一数据流分别来自 K1 个第一数据流分组。

在一些可能的实施方式中，卷积交织器具体用于：根据 p 条延迟线对 1 条通道数据流进行延迟得到 1 条第一数据流，p 为大于 1 的整数,p*a≥F/k2，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 d 个符号，z=p*d。每条通道数据流中的符号按照 p 条延迟线的序号依次输入 p 条延迟线，每条延迟线单次输入 d 个符号且单次输出 d 个符号，1 条第一数据流中连续的 p*d 个符号包括每条延迟线输出的 d 个符号，Q 为大于或等于 1 的整数，d 为大于或等于 1 的整数，d≤a。

在一些可能的实施方式中，p 条延迟线中序号最大的延迟线包括 0 个存储单元，其中，$d(p*Q+1)\geq K4$。

在一些可能的实施方式中，p 条延迟线中序号最小的延迟线包括 0 个存储单元，其中，$d(p*Q-1)\geq K4$。

在一些可能的实施方式中，参与分组交织的 K 条第一数据流包括第一符号矩阵，第一符号矩阵包括 K 行 B 列个符号，B=R*p*d，R 为大于或等于 1 的整数，经过分组交织后的 S 条目标数据流包括第二符号矩阵，第二符号矩阵包括 S 行 F 列个符号，K*B=S*F。第一符号矩阵中的符号至少来自 F 个不同码字，且第一符号矩阵中最多有 R*K1/K2 个符号来自相同的码字。

在一些可能的实施方式中，第二符号矩阵中每行的 F 个符号来自第一符号矩阵中至少 $\lfloor F/K2\rfloor$列，且$\lfloor F/K2\rfloor$列中每列最多取$K2$个符号，$\lfloor F/K2\rfloor$表示对 $F/K2$的商进行向上取整的整数。第二符号矩阵中每行的 F 个符号包括第一符号矩阵中每行的至少$\lfloor F/K1\rfloor$个符号，$\lfloor F/K1\rfloor$表示对 F/K1 的商进行向下取整的整数，且第二符号矩阵中每行的 F 个符号包括第一符号矩阵中每行的最多$\lfloor F/K1\rfloor$个符号，$\lfloor F/K1\rfloor$表示对 F/K 的商进行向上取整的整数。

在一些可能的实施方式中，第二符号矩阵的每一行中来自第一符号矩阵奇数列的符号分别位于第一符号矩阵的不同行，第二符号矩阵的每一行中来自第一符号矩阵偶数列的符号位于第一符号矩阵的不同行。

在一些可能的实施方式中，第二符号矩阵的每一行中由延迟取值相同的延迟线输出的符号来自第一符号矩阵的不同行。

在一些可能的实施方式中，第二符号矩阵的每一行的最多 K3 个符号来自第一符号矩阵的同一行，K3 个符号中任意 2 个符号分别由延迟差值大于或等于 2*Q*d 的 2 条延迟线输出。

在一些可能的实施方式中，卷积交织模块包括第一分组交织器、卷积交织器和第二分组

交织器。第一分组交织器用于：对 n 条通道数据流进行第一分组交织得到 T 条第一数据流，每条第一数据流中连续的 C 个符号至少来自 E 个不同的码字，T=n/K1，C 为 a 的倍数，E≥K2*a。卷积交织器用于：对 T 条第一数据流进行卷积交织得到 T 条第二数据流，每条第二数据流中连续的 H 个符号来自至少 F 个不同码字，F≥E，每条第二数据流中连续的 H 个符号最多 K1/K2 个符号来自相同码字。第二分组交织器用于：对 T 条第二数据流中每条第二数据流进行第二分组交织得到 S 条目标数据流，以得到共 m 条目标数据流，m=T*S，S≥k1/K2。

在一些可能的实施方式中，参与第一分组交织的 n 条通道数据流包括第三符号矩阵，第三符号矩阵包括 n 行 A 列个符号，A 为 a 的倍数，经过第一分组交织后的 T 条第一数据流包括第四符号矩阵，第四符号矩阵包括 T 行 C 列个符号，T 为 n 的约数，n*A=T*C。第三符号矩阵的一列中每连续 T 个符号为符号子矩阵，第四符号矩阵中每一列的 T 个符号分别与第三符号矩阵中每个符号子矩阵一一对应。

在一些可能的实施方式中，第三符号矩阵中的符号子矩阵按第一顺序排列，第三符号矩阵中每列的第 1 行到第 n 行包括按第一顺序排列的第 1 个符号子矩阵到第 n/T 个符号子矩阵，第三符号矩阵的相邻两列中前一列的第 n/T 个符号子矩阵到后一列的第 1 个符号子矩阵为按第一顺序排列的连续 2 个符号子矩阵，第四符号矩阵中第 1 列的 T 个符号来自第三符号矩阵中按第一顺序排列的第一个符号子矩阵，依次类推，直到第四符号矩阵中第 C 列的 T 个符号来自第三符号矩阵中按第一顺序排列的最后一个符号子矩阵。或者，第三符号矩阵中的符号子矩阵按第二顺序排列，第三符号矩阵中每 T 行的第 1 列到第 A 行包括按第二顺序排列的第 1 个符号子矩阵到第 A 个符号子矩阵，第三符号矩阵的相邻两个连续 T 行中前 T 行的第 A 个符号子矩阵到后 T 行的第 1 个符号子矩阵为按第二顺序排列的连续 2 个符号子矩阵，第四符号矩阵中第 1 列的 T 个符号来自第三符号矩阵中按第二顺序排列的第一个符号子矩阵，依次类推，直到第四符号矩阵中第 C 列的 T 个符号来自第三符号矩阵中按第二顺序排列的最后一个符号子矩阵。

在一些可能的实施方式中，卷积交织器具体用于：根据 p 条延迟线对 1 条第一数据流进行延迟得到 1 条第二数据流，p 为大于 1 的整数，p*E≥F，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 C 个符号，p*C=H。每条第一数据流中的符号按照 p 条延迟线的序号依次输入 p 条延迟线，每条延迟线单次输入 C 个符号且单次输出 C 个符号，1 条第二数据流中连续的 p*C 个符号包括每条延迟线输出的 C 个符号，Q 为大于或等于 1 的整数。

在一些可能的实施方式中，p 条延迟线中序号最大的延迟线包括 0 个存储单元，其中，$C(p*Q+1)\geq K1*K4$。

在一些可能的实施方式中，p 条延迟线中序号最小的延迟线包括 0 个存储单元，其中，$C(p*Q-1)\geq K1*K4$。

在一些可能的实施方式中，每条第二数据流包括 R 个符号集合，每个符号集合包括 p 个符号子集，每个符号子集包括 C 个符号，p 个符号子集分别由 p 条延迟线输出，每个符号集合中的符号来自至少 F 个不同码字，每条目标数据流包括 F 个符号，R*p*C=S*F，R 为大于或等于的整数。目标数据流中 F 个符号来自至少$\lfloor F/K2*a)\rfloor$个不同的符号子集，且$\lceil F/(K2*a)\rceil$

个不同的符号子集中每个符号子集最多取K2＊a个符号，⌈F/（K2＊a)⌉表示对F/（K2＊a)的商进行向上取整的整数。

在一些可能的实施方式中，目标数据流中F个符号包括来自第一符号子集中的第一组符号和来自第二符号子集的第二组符号，第一符号子集和第二符号子集属于同一个符号集合，第一符号子集和第二符号子集分别由两条相邻的延迟线输出，第一符号子集中的符号和第二符号子集中的符号分别按次序排列，第一组符号在第一符号子集中的次序与第二组符号在第二符号子集中的次序不同。目标数据流中F个符号包括来自第三符号子集中的第三组符号和来自第四符号子集的第四组符号，第三符号子集和第四符号子集属于不同的符号集合，第三符号子集和第四符号子集由同一条延迟线输出，第三符号子集中的符号和第四符号子集中的符号分别按次序排列，第三组符号在第三符号子集中的次序与第四组符号在第四符号子集中的次序不同。

在一些可能的实施方式中，每条目标数据流中F个符号最多有⌈F/R⌉个符号来自相同的符号集合，⌈F/R⌉表示对F/R的商进行向上取整的整数。

在一些可能的实施方式中，数据处理装置还包括编码器，得到共m条目标数据流之后，编码器用于：对m条目标数据流分别进行第二FEC编码，第二FEC编码的信息位长度等于F个符号。

第五方面，本申请提供了一种数据处理方法，该方法包括如下步骤。首先，对n条通道数据流中的每t条通道数据流进行分组交织得到s条第一数据流，以得到共m条第一数据流。n=q*t，m=q*s，所述n为大于1的整数，所述n能被所述q整除，所述q为大于或等于1的整数，所述t为大于或等于1的整数，所述s为大于或等于1的整数。所述n条通道数据流都经过第一前向纠错FEC编码，所述第一FEC编码后的每a个码字分布在b条所述通道数据流中，所述a≤所述b≤所述n，所述n能被所述b整除，所述a为大于或等于1的整数。每条通道数据流中每连续的a个符号来自a个不同的码字，且每条通道数据流中每连续的L1个符号来自至少a个不同的码字，L1=N*a/b，所述N表示所述码字的长度。所述t条通道数据流包括每条通道数据流中连续的a个符号共t*a个符号，所述t*a个符号包括每个符号中的Δ个比特共D个比特，D=Δ*t*a，所述D个比特在所述s条第一数据流中任一条第一数据流中连续，Δ=M/s，所述M表示一个符号包含的比特数。之后，对所述m条第一数据流分别进行卷积交织得到m条第二数据流。

在一些可能的实施方式中，每条所述第一数据流中每连续的d个符号来自v个不同的码字，且每条所述第一数据流中每连续的L2个符号来自至少v个不同的码字，所述v能被所述a整除，L2=t/s*L1，d=D/M。

在一些可能的实施方式中，所述n=32，所述n条通道数据流中位于奇数通道的16条通道数据流来自相同的码字，所述n条通道数据流中位于偶数通道的16条通道数据流来自相同的码字，所述n条通道数据流中位于奇数通道的数据流和位于偶数通道的数据流来自不同的码字。

在一些可能的实施方式中，所述t=2，所述s=1，对n条通道数据流中的每t条通道数据

流进行分组交织得到 s 条第一数据流包括：对第 2*i 条通道数据流和第 2*i+1 条通道数据流进行分组交织得到 1 条第一数据流，0≤i＜16。所述第 2*i 条通道数据流中连续的 2 个符号和所述第 2*i+1 条通道数据流中连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

在一些可能的实施方式中，所述分组交织得到的 1 条第一数据流中连续的 40 个比特中第 j 个比特来自第$2*i+\left\lfloor\frac{j}{\beta}\right\rfloor\%2$条通道数据流中连续 20 个比特中的第$\left\lfloor\left\lfloor\frac{j}{\beta}\right\rfloor/2\right\rfloor*\beta+j\%\beta$个比特，所述$\lfloor x\rfloor$表示向下取整，$0\le j<40$，所述β为 1, 2, 4, 5, 10 或 20。

在一些可能的实施方式中，所述 t=2，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：对第 2*i 条通道数据流和第 2*i+1 条通道数据流进行分组交织得到 1 条第一数据流，0≤i＜16。所述第 2*i 条通道数据流中第 j 组连续的β个比特与所述第 2*i+1 条通道数据流中第 j 组连续的β个比特在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述β为 1, 2, 4, 5, 10 或 20，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

在一些可能的实施方式中，所述 t=2，所述 s=2，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：对第 2*i 条通道数据流和第 2*i+1 条通道数据流进行分组交织得到第 2*i 条第一数据流和第 2*i+1 条第一数据流，0≤i＜16。所述第 2*i 条通道数据流中连续的 2 个符号和所述第 2*i+1 条通道数据流中连续的 2 个符号共 4 个符号中每个符号的其中 5 个比特共 20 个比特在所述第 2*i 条第一数据流中连续，所述第 2*i 条第一数据流中每连续的 20 个比特来自 4 个不同的码字，所述第 2*i 条通道数据流中连续的 2 个符号和所述第 2*i+1 条通道数据流中连续的 2 个符号共 4 个符号中每个符号的另外 5 个比特共 20 个比特在所述第 2*i+1 条第一数据流中连续，所述第 2*i+1 条第一数据流中每连续的 20 个比特来自 4 个不同的码字。

在一些可能的实施方式中，第 2*i+g 条第一数据流中连续的 20 个比特中第 f 个比特来自第$2*i+\left\lfloor\frac{f}{10}\right\rfloor$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f\%10}{5}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

在一些可能的实施方式中，第 2*i+g 条第一数据流中连续的 20 个比特中第 f 个比特来自第$2*i+\left\lfloor\frac{f}{5}\right\rfloor\%2$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f}{10}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

在一些可能的实施方式中，所述 t=4，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流和第 4*i+3 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的连续 2 符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自至少 4 个不同的码字、且每连续的 272 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的

8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 t=4，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流和第 4*i+3 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，所述 4 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自至少 4 个不同的码字、且每连续的 272 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 t=4，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流和第 4*i+3 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的第 j 个符号共 4 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

在一些可能的实施方式中，所述 t=8，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：对第 8*i 条通道数据流、第 8*i+1 条通道数据流、第 8*i+2 条通道数据流、第 8*i+3 条通道数据流、第 8*i+4 条通道数据流、第 8*i+5 条通道数据流、第 8*i+6 条通道数据流、第 8*i+7 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的连续 2 符号共 16 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号来自至少 4 个不同的码字、且每连续的 544 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 8 个符号、第 9 个符号、第 10 个符号和第 11 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 12 个符号、第 13 个符号、第 14 个符号和第 15 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 t=8，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 8*i 条通道数据流、第 8*i+1 条通道数据流、第 8*i+2 条通道数据流、第 8*i+3 条通道数据流、第 8*i+4 条通道数据流、第 8*i+5 条通道数据流、第 8*i+6 条通道数据流、第 8*i+7 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8

条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号共 16 个符号在所述分组交织得到的 1 条第一数据流中连续，所述 8 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号来自至少 4 个不同的码字、且每连续的 544 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 8 个符号、第 9 个符号、第 10 个符号和第 11 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 12 个符号、第 13 个符号、第 14 个符号和第 15 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 t=8，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：对第 8*i 条通道数据流、第 8*i+1 条通道数据流、第 8*i+2 条通道数据流、第 8*i+3 条通道数据流、第 8*i+4 条通道数据流、第 8*i+5 条通道数据流、第 8*i+6 条通道数据流、第 8*i+7 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的第 j 个符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 n=32，所述 n 条通道数据流中排序在前的连续 16 条通道数据流来自相同的码字，所述 n 条通道数据流中排序在后的连续 16 条通道数据流来自相同的码字，所述 n 条通道数据流中排序在前的连续 16 条通道数据流和排序在后的连续 16 条通道数据流来自不同的码字。

在一些可能的实施方式中，所述 t=2，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：对第 i 条通道数据流和第 i+16 条通道数据流进行分组交织得到 1 条第一数据流，0≤i＜16。所述第 i 条通道数据流中连续的 2 个符号和所述第 i+16 条通道数据流中连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

在一些可能的实施方式中，所述分组交织得到的 1 条第一数据流中连续的 40 个比特中第 j 个比特来自第 $2*i+\left\lfloor \frac{j}{\beta} \right\rfloor \%2$ 条通道数据流中连续 20 个比特中的第 $\left\lfloor \left\lfloor \frac{j}{\beta} \right\rfloor /2 \right\rfloor *\beta+j\%\beta$ 个比特，所述 $\lfloor x \rfloor$ 表示向下取整，$0 \le j < 40$，所述 β 为 1, 2, 4, 5, 10 或 20。

在一些可能的实施方式中，所述 t=2，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：对第 i 条通道数据流和第 i+16 条通道数据流进行分组交织得到 1 条第一数据流，0≤i＜16。所述第 i 条通道数据流中第 j 组连续的 β 个比特与

所述第 i+16 条通道数据流中第 j 组连续的β个比特在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述β为 1, 2, 4, 5, 10 或 20，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

在一些可能的实施方式中，所述 t=2，所述 s=2，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：对第 i 条通道数据流和第 i+16 条通道数据流进行分组交织得到第 2*i 条第一数据流和第 2*i+1 条第一数据流，0≤i＜16。所述第 i 条通道数据流中连续的 2 个符号和所述第 i+16 条通道数据流中连续的 2 个符号共 4 个符号中每个符号的其中 5 个比特共 20 个比特在所述第 2*i 条第一数据流中连续，所述第 2*i 条第一数据流中每连续的 20 个比特来自 4 个不同的码字，所述第 i 条通道数据流中连续的 2 个符号和所述第 i+16 条通道数据流中连续的 2 个符号共 4 个符号中每个符号的另外 5 个比特共 20 个比特在所述第 2*i+1 条第一数据流中连续，所述第 2*i+1 条第一数据流中每连续的 20 个比特来自 4 个不同的码字。

在一些可能的实施方式中，第 2*i+g 条第一数据流中连续的 20 个比特中第f个比特来自第$i+16*\left\lfloor\frac{f}{10}\right\rfloor$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f\%10}{5}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

在一些可能的实施方式中，第 2*i+g 条第一数据流中连续的 20 个比特中第f个比特来自第$i+16*\left\lfloor\frac{f}{10}\right\rfloor$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f}{10}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

在一些可能的实施方式中，所述 t=4，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：对第 2*i 条通道数据流、第 2*i+1 条通道数据流、第 2*i+16 条通道数据流和第 2*i+17 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的连续 2 符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自至少 4 个不同的码字、且每连续的 272 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 t=4，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：对第 2*i 条通道数据流、第 2*i+1 条通道数据流、第 2*i+16 条通道数据流和第 2*i+17 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，所述 4 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自至少 4 个不同的码字、且每连续的 272 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6

个符号和第 7 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 t=4，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：对第 2*i 条通道数据流、第 2*i+1 条通道数据流、第 2*i+16 条通道数据流和第 2*i+17 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的第 j 个符号共 4 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

在一些可能的实施方式中，所述 t=8，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流、第 4*i+16 条通道数据流、第 4*i+17 条通道数据流、第 4*i+18 条通道数据流和第 4*i+19 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的连续 2 符号共 16 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号来自至少 4 个不同的码字、且每连续的 544 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 8 个符号、第 9 个符号、第 10 个符号和第 11 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 12 个符号、第 13 个符号、第 14 个符号和第 15 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 t=8，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流、第 4*i+16 条通道数据流、第 4*i+17 条通道数据流、第 4*i+18 条通道数据流和第 4*i+19 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号共 16 个符号在所述分组交织得到的 1 条第一数据流中连续，所述 8 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号来自至少 4 个不同的码字、且每连续的 544 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 8 个符号、第 9 个符号、第 10 个符号和第 11 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 12 个符号、第 13 个符号、第 14 个符号和第 15 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 t=8，所述 s=1，对 n 条通道数据流中的每 t 条通道数据

流进行分组交织得到 s 条第一数据流包括：对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流、第 4*i+16 条通道数据流、第 4*i+17 条通道数据流、第 4*i+18 条通道数据流和第 4*i+19 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的第 j 个符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号至少来自 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

在一些可能的实施方式中，对 1 条第一数据流进行卷积交织得到 1 条第二数据流包括：根据 p 条延迟线对 1 条第一数据流进行延迟得到 1 条第二数据流。所述 p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 d 个符号，每条通道数据流中的符号按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 d 个符号且单次输出 d 个符号，1 条所述第二数据流中连续的 p*d 个符号包括每条延迟线输出的 d 个符号，所述 Q 为大于或等于 1 的整数。

在一些可能的实施方式中，所述 p 条延迟线中序号最大的延迟线包括 0 个存储单元，$d(p*Q+1) \geq L2$，L2=t/s*L1。或者，所述 p 条延迟线中序号最小的延迟线包括 0 个存储单元，$d(p*Q-1) \geq L2$，L2=t/s*L1。

在一些可能的实施方式中，对所述 m 条第一数据流分别进行卷积交织得到 m 条第二数据流之后，所述方法还包括：对所述 m 条第二数据流分别进行第二 FEC 编码得到 m 条编码数据流。每条所述编码数据流中长度为 K 个符号的信息数据最多来自 K 个不同的码字，K≥p*d。

在一些可能的实施方式中，对 1 条第一数据流进行卷积交织得到 1 条第二数据流包括：根据 p 条延迟线对 1 条第一数据流进行延迟得到 1 条第二数据流，所述 p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 4 个符号，每条通道数据流中的符号按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 4 个符号且单次输出 4 个符号，1 条所述第二数据流中连续的 p*4 个符号包括每条延迟线输出的 4 个符号，所述 Q 满足$4(p*Q-1) \geq 272$ 或者 $4(p*Q+1) \geq 272$ 或者 $4(p*Q-1) \geq 544$ 或者 $4(p*Q+1) \geq 544$。

在一些可能的实施方式中，对所述 m 条第一数据流分别进行卷积交织得到 m 条第二数据流之后，所述方法还包括：对所述 m 条第二数据流分别进行第二 FEC 编码得到 m 条编码数据流，每条所述编码数据流中长度为 K 个符号的信息数据最多来自 K 个不同的码字，K≥p*4。

在一些可能的实施方式中，对 1 条第一数据流进行卷积交织得到 1 条第二数据流包括：根据 p 条延迟线对 1 条第一数据流进行延迟得到 1 条第二数据流，所述 p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q；每个存储单元用于存储 34 个比特，每条通道

数据流中的比特按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 34 个比特且单次输出 34 个比特，1 条所述第二数据流中连续的 p*34 个比特包括每条延迟线输出的 34 个比特；或者，每个存储单元用于存储 68 个比特，每条通道数据流中的比特按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 68 个比特且单次输出 68 个比特，1 条所述第二数据流中连续的 p*68 个比特包括每条延迟线输出的 68 个比特。

在一些可能的实施方式中，p=2，每个存储单元用于存储 68 个比特，或者，p=4，每个存储单元用于存储 34 个比特。

第六方面，本申请提供了一种数据处理装置。该数据处理装置包括：分组交织器和卷积交织器。所述分组交织器用于：对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流，以得到共 m 条第一数据流。n=q*t，m=q*s，所述 n 为大于 1 的整数，所述 n 能被所述 q 整除，所述 q 为大于或等于 1 的整数，所述 t 为大于或等于 1 的整数，所述 s 为大于或等于 1 的整数。所述 n 条通道数据流都经过第一前向纠错 FEC 编码，所述第一 FEC 编码后的每 a 个码字分布在 b 条所述通道数据流中，所述 a≤所述 b≤所述 n，所述 n 能被所述 b 整除，所述 a 为大于或等于 1 的整数。每条通道数据流中每连续的 a 个符号来自 a 个不同的码字，且每条通道数据流中每连续的 L1 个符号来自至少 a 个不同的码字，L1=N*a/b,所述 N 表示所述码字的长度。所述 t 条通道数据流包括每条通道数据流中连续的 a 个符号共 t*a 个符号，所述 t*a 个符号包括每个符号中的 Δ 个比特共 D 个比特，D=Δ*t*a，所述 D 个比特在所述 s 条第一数据流中任一条第一数据流中连续，Δ=M/s,所述 M 表示一个符号包含的比特数。所述卷积交织器用于：对所述 m 条第一数据流分别进行卷积交织得到 m 条第二数据流。

在一些可能的实施方式中，每条所述第一数据流中每连续的 d 个符号来自 v 个不同的码字，且每条所述第一数据流中每连续的 L2 个符号来自至少 v 个不同的码字，所述 v 能被所述 a 整除，L2=t/s*L1，d=D/M。

在一些可能的实施方式中，所述 n=32，所述 n 条通道数据流中位于奇数通道的 16 条通道数据流来自相同的码字，所述 n 条通道数据流中位于偶数通道的 16 条通道数据流来自相同的码字，所述 n 条通道数据流中位于奇数通道的数据流和位于偶数通道的数据流来自不同的码字。

在一些可能的实施方式中，所述 t=2，所述 s=1，所述分组交织器具体用于：对第 2*i 条通道数据流和第 2*i+1 条通道数据流进行分组交织得到 1 条第一数据流，0≤i＜16。所述第 2*i 条通道数据流中连续的 2 个符号和所述第 2*i+1 条通道数据流中连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

在一些可能的实施方式中，所述分组交织得到的 1 条第一数据流中连续的 40 个比特中第 j个比特来自第$2*i+\left\lfloor \frac{j}{\beta} \right\rfloor \%2$条通道数据流中连续 20 个比特中的第$\left\lfloor \left\lfloor \frac{j}{\beta} \right\rfloor /2 \right\rfloor *\beta+j\%\beta$个比特，所述$\lfloor x \rfloor$表示向下取整，$0 \le j < 40$，所述β为 1,2,4,5,10 或 20。

在一些可能的实施方式中，所述 t=2，所述 s=1，所述分组交织器具体用于：对第 2*i 条

通道数据流和第 2*i+1 条通道数据流进行分组交织得到 1 条第一数据流，0≤i＜16。所述第 2*i 条通道数据流中第 j 组连续的β个比特与所述第 2*i+1 条通道数据流中第 j 组连续的β个比特在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述β为 1,2,4,5,10 或 20，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

在一些可能的实施方式中，所述 t=2，所述 s=2，所述分组交织器具体用于：对第 2*i 条通道数据流和第 2*i+1 条通道数据流进行分组交织得到第 2*i 条第一数据流和第 2*i+1 条第一数据流，0≤i＜16。所述第 2*i 条通道数据流中连续的 2 个符号和所述第 2*i+1 条通道数据流中连续的 2 个符号共 4 个符号中每个符号的其中 5 个比特共 20 个比特在所述第 2*i 条第一数据流中连续，所述第 2*i 条第一数据流中每连续的 20 个比特来自 4 个不同的码字，所述第 2*i 条通道数据流中连续的 2 个符号和所述第 2*i+1 条通道数据流中连续的 2 个符号共 4 个符号中每个符号的另外 5 个比特共 20 个比特在所述第 2*i+1 条第一数据流中连续，所述第 2*i+1 条第一数据流中每连续的 20 个比特来自 4 个不同的码字。

在一些可能的实施方式中，第 2*i+g 条第一数据流中连续的 20 个比特中第f个比特来自第$2*i+\left\lfloor\frac{f}{10}\right\rfloor$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f\%10}{5}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

在一些可能的实施方式中，第 2*i+g 条第一数据流中连续的 20 个比特中第f个比特来自第$2*i+\left\lfloor\frac{f}{5}\right\rfloor\%2$ 条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f}{10}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

在一些可能的实施方式中，所述 t=4，所述 s=1，所述分组交织器具体用于：对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流和第 4*i+3 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的连续 2 符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自至少 4 个不同的码字、且每连续的 272 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 t=4，所述 s=1，所述分组交织器具体用于：对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流和第 4*i+3 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，所述 4 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自至少 4 个不同的码字、且每连续的 272 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 t=4，所述 s=1，所述分组交织器具体用于：对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流和第 4*i+3 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的第 j 个符号共 4 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

在一些可能的实施方式中，所述 t=8，所述 s=1，所述分组交织器具体用于：对第 8*i 条通道数据流、第 8*i+1 条通道数据流、第 8*i+2 条通道数据流、第 8*i+3 条通道数据流、第 8*i+4 条通道数据流、第 8*i+5 条通道数据流、第 8*i+6 条通道数据流、第 8*i+7 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的连续 2 符号共 16 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号来自至少 4 个不同的码字、且每连续的 544 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 8 个符号、第 9 个符号、第 10 个符号和第 11 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 12 个符号、第 13 个符号、第 14 个符号和第 15 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 t=8，所述 s=1，所述分组交织器具体用于：对第 8*i 条通道数据流、第 8*i+1 条通道数据流、第 8*i+2 条通道数据流、第 8*i+3 条通道数据流、第 8*i+4 条通道数据流、第 8*i+5 条通道数据流、第 8*i+6 条通道数据流、第 8*i+7 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号共 16 个符号在所述分组交织得到的 1 条第一数据流中连续，所述 8 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号来自至少 4 个不同的码字、且每连续的 544 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 8 个符号、第 9 个符号、第 10 个符号和第 11 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 12 个符号、第 13 个符号、第 14 个符号和第 15 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 t=8，所述 s=1，所述分组交织器具体用于：对第 8*i 条通道数据流、第 8*i+1 条通道数据流、第 8*i+2 条通道数据流、第 8*i+3 条通道数据流、第 8*i+4 条通道数据流、第 8*i+5 条通道数据流、第 8*i+6 条通道数据流、第 8*i+7 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的第 j 个符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，

j≥0，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 n=32，所述 n 条通道数据流中排序在前的连续 16 条通道数据流来自相同的码字，所述 n 条通道数据流中排序在后的连续 16 条通道数据流来自相同的码字，所述 n 条通道数据流中排序在前的连续 16 条通道数据流和排序在后的连续 16 条通道数据流来自不同的码字。

在一些可能的实施方式中，所述 t=2，所述 s=1，所述分组交织器具体用于：对第 i 条通道数据流和第 i+16 条通道数据流进行分组交织得到 1 条第一数据流，0≤i＜16。所述第 i 条通道数据流中连续的 2 个符号和所述第 i+16 条通道数据流中连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

在一些可能的实施方式中，所述分组交织得到的 1 条第一数据流中连续的 40 个比特中第 j个比特来自第$2*i+\left\lfloor\frac{j}{\beta}\right\rfloor\%2$条通道数据流中连续 20 个比特中的第$\left\lfloor\left\lfloor\frac{j}{\beta}\right\rfloor/2\right\rfloor*\beta+j\%\beta$个比特，所述$[x]$表示向下取整，$0\le j<40$，所述$\beta$为 1, 2, 4, 5, 10 或 20。

在一些可能的实施方式中，所述 t=2，所述 s=1，所述分组交织器具体用于：对第 i 条通道数据流和第 i+16 条通道数据流进行分组交织得到 1 条第一数据流，0≤i＜16。所述第 i 条通道数据流中第 j 组连续的β个比特与所述第 i+16 条通道数据流中第 j 组连续的β个比特在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述β为 1, 2, 4, 5, 10 或 20，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

在一些可能的实施方式中，所述 t=2，所述 s=2，所述分组交织器具体用于：对第 i 条通道数据流和第 i+16 条通道数据流进行分组交织得到第 2*i 条第一数据流和第 2*i+1 条第一数据流，0≤i＜16。所述第 i 条通道数据流中连续的 2 个符号和所述第 i+16 条通道数据流中连续的 2 个符号共 4 个符号中每个符号的其中 5 个比特共 20 个比特在所述第 2*i 条第一数据流中连续，所述第 2*i 条第一数据流中每连续的 20 个比特来自 4 个不同的码字，所述第 i 条通道数据流中连续的 2 个符号和所述第 i+16 条通道数据流中连续的 2 个符号共 4 个符号中每个符号的另外 5 个比特共 20 个比特在所述第 2*i+1 条第一数据流中连续，所述第 2*i+1 条第一数据流中每连续的 20 个比特来自 4 个不同的码字。

在一些可能的实施方式中，第 2*i+g 条第一数据流中连续的 20 个比特中第f个比特来自第$i+16*\left\lfloor\frac{f}{10}\right\rfloor$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f\%10}{5}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

在一些可能的实施方式中，第 2*i+g 条第一数据流中连续的 20 个比特中第f个比特来自第$i+16*\left\lfloor\frac{f}{10}\right\rfloor$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f}{10}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

在一些可能的实施方式中，所述 t=4，所述 s=1，所述分组交织器具体用于：对第 2*i 条

通道数据流、第 2*i+1 条通道数据流、第 2*i+16 条通道数据流和第 2*i+17 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的连续 2 符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自至少 4 个不同的码字、且每连续的 272 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 t=4，所述 s=1，所述分组交织器具体用于：对第 2*i 条通道数据流、第 2*i+1 条通道数据流、第 2*i+16 条通道数据流和第 2*i+17 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，所述 4 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自至少 4 个不同的码字、且每连续的 272 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 t=4，所述 s=1，所述分组交织器具体用于：对第 2*i 条通道数据流、第 2*i+1 条通道数据流、第 2*i+16 条通道数据流和第 2*i+17 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的第 j 个符号共 4 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

在一些可能的实施方式中，所述 t=8，所述 s=1，所述分组交织器具体用于：对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流、第 4*i+16 条通道数据流、第 4*i+17 条通道数据流、第 4*i+18 条通道数据流和第 4*i+19 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的连续 2 符号共 16 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号来自至少 4 个不同的码字、且每连续的 544 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 8 个符号、第 9 个符号、第 10 个符号和第 11 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 12 个符号、第 13 个符号、第 14 个符号和第 15 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 t=8，所述 s=1，所述分组交织器具体用于：对第 4*i 条

通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流、第 4*i+16 条通道数据流、第 4*i+17 条通道数据流、第 4*i+18 条通道数据流和第 4*i+19 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号共 16 个符号在所述分组交织得到的 1 条第一数据流中连续，所述 8 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的16个符号来自至少4个不同的码字、且每连续的544个符号来自至少4个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 8 个符号、第 9 个符号、第 10 个符号和第 11 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 12 个符号、第 13 个符号、第 14 个符号和第 15 个符号分别来自不同的码字。

在一些可能的实施方式中，所述 t=8，所述 s=1，所述分组交织器具体用于：对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流、第 4*i+16 条通道数据流、第 4*i+17 条通道数据流、第 4*i+18 条通道数据流和第 4*i+19 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的第 j 个符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号至少来自 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

在一些可能的实施方式中，所述卷积交织器具体用于：根据 p 条延迟线对 1 条第一数据流进行延迟得到 1 条第二数据流，所述 p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 d 个符号，每条通道数据流中的符号按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 d 个符号且单次输出 d 个符号，1 条所述第二数据流中连续的 p*d 个符号包括每条延迟线输出的 d 个符号，所述 Q 为大于或等于 1 的整数。

在一些可能的实施方式中，所述 p 条延迟线中序号最大的延迟线包括 0 个存储单元，$d(p*Q+1) \geq L2$，L2=t/s*L1。或者，所述 p 条延迟线中序号最小的延迟线包括 0 个存储单元，$d(p*Q-1) \geq L2$，L2=t/s*L1。

在一些可能的实施方式中，所述数据处理装置还包括编码器，得到 m 条第二数据流之后，所述编码器用于：对所述 m 条第二数据流分别进行第二 FEC 编码得到 m 条编码数据流。每条所述编码数据流中长度为 K 个符号的信息数据最多来自 K 个不同的码字，K≥p*d。

在一些可能的实施方式中，所述卷积交织器具体用于：根据 p 条延迟线对 1 条第一数据

流进行延迟得到 1 条第二数据流，所述 p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 4 个符号，每条通道数据流中的符号按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 4 个符号且单次输出 4 个符号，1 条所述第二数据流中连续的 p*4 个符号包括每条延迟线输出的 4 个符号，所述 Q 满足 $4(p*Q-1)\geq 272$ 或者 $4(p*Q+1)\geq 272$ 或者 $4(p*Q-1)\geq 544$ 或者 $4(p*Q+1)\geq 544$。

在一些可能的实施方式中，所述数据处理装置还包括编码器，得到 m 条第二数据流之后，所述编码器用于：对所述 m 条第二数据流分别进行第二 FEC 编码得到 m 条编码数据流，每条所述编码数据流中长度为 K 个符号的信息数据最多来自 K 个不同的码字，K≥p*4。

在一些可能的实施方式中，所述卷积交织器具体用于：根据 p 条延迟线对 1 条第一数据流进行延迟得到 1 条第二数据流，所述 p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q。每个存储单元用于存储 34 个比特，每条通道数据流中的比特按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 34 个比特且单次输出 34 个比特，1 条所述第二数据流中连续的 p*34 个比特包括每条延迟线输出的 34 个比特；或者，每个存储单元用于存储 68 个比特，每条通道数据流中的比特按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 68 个比特且单次输出 68 个比特，1 条所述第二数据流中连续的 p*68 个比特包括每条延迟线输出的 68 个比特。

在一些可能的实施方式中，p=2，每个存储单元用于存储 68 个比特，或者，p=4，每个存储单元用于存储 34 个比特。

第七方面，本申请提供了一种数据处理方法，该方法包括如下步骤。首先，根据 p 条延迟线对 n 条通道数据流分别进行延迟得到 n 条第一数据流。其中，所述 n 条通道数据流都经过第一前向纠错 FEC 编码，所述 p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 U 个比特，每条通道数据流中的比特按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 U 个比特且单次输出 U 个比特，1 条所述第二数据流中连续的 p*U 个比特包括每条延迟线输出的 U 个比特，所述 Q 为大于或等于 1 的整数，所述 U 为大于或等于 1 的整数。进而，对所述 n 条第一数据流分别进行第二 FEC 编码得到 n 条第二数据流。其中，经过所述第二 FEC 编码后的所述第二数据流中每个码字的信息数据为所述 p 条延迟线单次输出的 p*U 个比特。

在一些可能的实施方式中，p*U=120、136 或 160。

第八方面，本申请提供了一种数据处理装置，该数据处理装置包括卷积交织器和编码器。所述卷积交织器用于：根据 p 条延迟线对 n 条通道数据流分别进行延迟得到 n 条第一数据流。其中，所述 n 条通道数据流都经过第一前向纠错 FEC 编码，所述 p 为大于 1 的整数，每条延

迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 U 个比特，每条通道数据流中的比特按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 U 个比特且单次输出 U 个比特，1 条所述第二数据流中连续的 p*U 个比特包括每条延迟线输出的 U 个比特，所述 Q 为大于或等于 1 的整数，所述 U 为大于或等于 1 的整数。所述编码器用于：对所述 n 条第一数据流分别进行第二 FEC 编码得到 n 条第二数据流。其中，经过所述第二 FEC 编码后的所述第二数据流中每个码字的信息数据为所述 p 条延迟线单次输出的 p*U 个比特。

在一些可能的实施方式中，p*U=120、136 或 160。

第九方面，本申请提供了一种计算机可读存储介质，计算机可读存储介质存储有计算机程序，其中，计算机程序被硬件执行时能够实现上述第一方面、第三方面、第五方面或第七方面中任意一种方法的部分或全部步骤。

本申请实施例中，n 路通道数据流都为经过外码编码后的码字流，在对 n 路数据流分别进行卷积交织处理，并对卷积交织后的 n 路数据流进行数据流复用为 m 路第二数据流，随后进行内码编码。采用本申请提供的数据交织和复用处理方案，通过较短的时延就能实现复用处理后的 m 路数据流在连续输出的多个符号来自多个不同的外码码字，使得级联 FEC 方案在保证较好性能的基础上也有助于降低数据交织的时延。也就是说，本申请中卷积交织处理与数据复用相结合的方案使得级联 FEC 方案的整体时延较低，更适用于要求低时延的应用场景。

附图说明

图 1 为本申请实施例应用的一种通信系统示意图；

图 2 为图 1 所示通信系统中一种数据传输的过程示意图；

图 3（a）为本申请实施例中发端处理模块的第一种数据处理示意图；

图 3（b）为本申请实施例中发端处理模块的第二种数据处理示意图；

图 3（c）为本申请实施例中发端处理模块的第三种数据处理示意图；

图 3（d）为本申请实施例中发端处理模块的第四种数据处理示意图；

图 3（e）为本申请实施例中通道数据对齐处理的示意图；

图 3（f）为本申请实施例中发端处理模块的第五种数据处理示意图；

图 3（g）为本申请实施例中发端处理模块的第六种数据处理示意图；

图 3（h）为本申请实施例中发端处理模块的第七种数据处理示意图；

图 3（i）为本申请实施例中发端处理模块的第八种数据处理示意图；

图 4（a）为本申请实施例中针对收端处理模块的第一种数据处理示意图；

图 4（b）为本申请实施例中针对收端处理模块的第二种数据处理示意图；

图 4（c）为本申请实施例中针对收端处理模块的第三种数据处理示意图；

图 5 为发端设备采用 1×800G 接口的 32 条 PCS 通道数据流的示意图；

图 6 为发端设备采用 2×400G 接口的 32 条 PCS 通道数据流的示意图；

图 7 为发端设备采用 4×200G 接口的 32 条 PCS 通道数据流的示意图；
图 8 为发端设备采用 8×100G 接口的 32 条 FEC 通道数据流的一种示意图；
图 9 为发端设备采用 8×100G 接口的 32 条 FEC 通道数据流的另一种示意图；
图 10 为本申请实施例提供的数据处理方法的一种流程示意图；
图 11 为本申请实施例中对 n 条通道数据流分别进行卷积交织的一种结构示意图；
图 12（a）为本申请实施例中卷积交织器的第一种结构示意图；
图 12（b）为本申请实施例中卷积交织器的第二种结构示意图；
图 13 为本申请实施例中对 n 条第一数据流进行复用的一种结构示意图；
图 14 为本申请实施例中复用器的第一种结构示意图；
图 15 为本申请实施例中对 m 条第二数据流进行 FEC 编码的一种结构示意图；
图 16（a）为本申请实施例中卷积交织器的第三种结构示意图；
图 16（b）为本申请实施例中卷积交织器的第四种结构示意图；
图 17（a）为本申请实施例中复用器的第二种结构示意图；
图 17（b）为本申请实施例中复用器的第三种结构示意图；
图 17（c）为本申请实施例中复用器的第四种结构示意图；
图 18（a）为本申请实施例中卷积交织器的第五种结构示意图；
图 18（b）为本申请实施例中卷积交织器的第六种结构示意图；
图 19（a）为本申请实施例中卷积交织器的第七种结构示意图；
图 19（b）为本申请实施例中卷积交织器的第八种结构示意图；
图 20 为本申请实施例中卷积交织器的第九种结构示意图；
图 21 为本申请实施例中卷积交织器的第十种结构示意图；
图 22 为本申请实施例中复用器的第五种结构示意图；
图 23 为本申请实施例中卷积交织器的第十一种结构示意图；
图 24 为本申请实施例中复用器的第六种结构示意图；
图 25 为本申请实施例中卷积交织器的第十二种结构示意图；
图 26 为本申请实施例中卷积交织器的第十三种结构示意图；
图 27（a）为本申请实施例中卷积交织器的第十四种结构示意图；
图 27（b）为本申请实施例中卷积交织器的第十五种结构示意图；
图 28（a）为本申请实施例中卷积交织器的第十六种结构示意图；
图 28（b）为本申请实施例中卷积交织器的第十七种结构示意图；
图 29（a）为本申请实施例中卷积交织器的第十八种结构示意图；
图 29（b）为本申请实施例中卷积交织器的第十九种结构示意图；
图 30（a）为本申请实施例中卷积交织器的第二十种结构示意图；
图 30（b）为本申请实施例中卷积交织器的第二十一种结构示意图；
图 31（a）为本申请实施例中卷积交织器的第二十二种结构示意图；
图 31（b）为本申请实施例中卷积交织器的第二十三种结构示意图；
图 32（a）为本申请实施例中对 n 条第一数据流进行分组交织的一种结构示意图；

图 32（b）为本申请实施例中分组交织器的一种结构示意图；
图 33 为本申请实施例中数据处理装置的一种结构示意图；
图 34 为本申请实施例提供的交织处理的一种流程示意图；
图 35 为本申请实施例中对 n 条第一数据流进行分组交织的一种结构示意图；
图 36 为本申请实施例中进行分组交织的一种实现方式示意图；
图 37 为客户侧接口为 2×400GbE 时通道对齐后数据流格式的示意图；
图 38 为分组交织的一种实现方式示意图；
图 39 为分组交织的另一种实现方式示意图；
图 40 为分组交织的另一种实现方式示意图；
图 41 为分组交织的另一种实现方式示意图；
图 42 为本申请实施例提供的交织处理的另一种流程示意图；
图 43 为本申请实施例中进行第一分组交织的一种实现方式示意图；
图 44（a）为本申请实施例中进行第二分组交织的一种实现方式示意图；
图 44（b）为本申请实施例中进行第二分组交织的一种具体实现方式示意图；
图 45（a）为第一分组交织的一种实现方式示意图；
图 45（b）为卷积交织的一种实现方式示意图；
图 45（c）为卷积交织的另一种实现方式示意图；
图 45（d）为第二分组交织的一种实现方式示意图；
图 46（a）为卷积交织的另一种实现方式示意图；
图 46（b）为卷积交织的另一种实现方式示意图；
图 46（c）为第二分组交织的另一种实现方式示意图；
图 47 为第二分组交织的另一种实现方式示意图；
图 48 为第二分组交织的另一种实现方式示意图；
图 49 为第一分组交织的另一种实现方式示意图；
图 50（a）为本申请实施例中数据处理装置的另一种结构示意图；
图 50（b）为本申请实施例中数据处理装置的另一种结构示意图；
图 51 为本申请实施例提供的数据处理方法的另一种流程示意图；
图 52 为本申请实施例中对 n 条通道数据流进行分组交织的一种结构示意图；
图 53 为本申请实施例中分组交织的一种应用场景示意图；
图 54 为本申请实施例中分组交织的几种具体实施例示意图；
图 55 为本申请实施例中分组交织的另一种应用场景示意图；
图 56 为本申请实施例中分组交织的几种具体实施例示意图；
图 57 为本申请实施例中分组交织的另一种应用场景示意图；
图 58 为本申请实施例中分组交织的另一种应用场景示意图；
图 59 为本申请实施例中分组交织的另一种应用场景示意图；
图 60 为本申请实施例中分组交织的另一种应用场景示意图；
图 61 为本申请实施例中分组交织的再一种应用场景示意图；

图 62 为本申请实施例中分组交织的再一种应用场景示意图；
图 63 为本申请实施例中分组交织的再一种应用场景示意图；
图 64 为本申请实施例中分组交织的再一种应用场景示意图；
图 65 为本申请实施例中分组交织的再一种应用场景示意图；
图 66 为本申请实施例中分组交织的再一种应用场景示意图；
图 67 为本申请实施例中分组交织的再一种应用场景示意图；
图 68 为本申请实施例中分组交织的再一种应用场景示意图；
图 69 为本申请实施例中分组交织的再一种应用场景示意图；
图 70 为本申请实施例中分组交织的再一种应用场景示意图；
图 71 为本申请实施例中分组交织的再一种应用场景示意图；
图 72 为本申请实施例中分组交织的再一种应用场景示意图；
图 73 为本申请实施例中对 m 条第一数据流分别进行卷积交织的一种结构示意图；
图 74 为本申请实施例中卷积交织器的一种实施例示意图；
图 75 为本申请实施例中卷积交织器的另一种实施例示意图；
图 76 为本申请实施例中卷积交织器的另一种实施例示意图；
图 77 为本申请实施例中卷积交织器的另一种实施例示意图；
图 78 为本申请实施例中卷积交织器的另一种实施例示意图；
图 79 为本申请实施例中卷积交织器的另一种实施例示意图；
图 80 为本申请实施例中卷积交织器的另一种实施例示意图；
图 81 为本申请实施例中卷积交织器的另一种实施例示意图；
图 82 为本申请实施例中卷积交织器的另一种实施例示意图；
图 83 为本申请实施例中卷积交织器的另一种实施例示意图；
图 84 为本申请实施例中内码编码的一种实现方式示意图；
图 85 为本申请实施例中数据处理装置的一种结构示意图；
图 86 为本申请实施例提供的数据处理方法的另一种流程示意图；
图 87 为本申请实施例中数据处理装置的另一种结构示意图；
图 88 为本申请实施例中数据处理装置的另一种结构示意图。

具体实施方式

本申请实施例提供了一种数据处理方法及数据处理装置。可以在采用较低时延的场景下获得较好的级联 FEC 方案性能。需要说明的是，本申请说明书和权利要求书及上述附图中的术语“第一”、“第二”等用于区别类似的对象，而非限定特定的顺序或先后次序。应该理解，上述术语在适当情况下可以互换，以便在本申请描述的实施例能够以除了在本申请描述的内容以外的顺序实施。此外，术语“包括”和“具有”以及他们的任何变形，意图在于覆盖不排他的包含。例如，包含了一系列步骤或单元的过程、方法、系统、产品或设备不必限于清楚地列出的那些步骤或单元，而是可包括没有清楚地列出的或对于这些过程、方法、产

品或设备固有的其它步骤或单元。

图 1 为本申请实施例应用的一种通信系统示意图。如图 1 所示，该通信系统包括发端设备 01、发端处理模块 02、信道传输媒介 03、收端处理模块 04 和收端设备 05。以该通信系统是数据中心网络为例，发端设备 01 和收端设备 05 可以为交换机或路由器等设备，且发端设备 01 也称为位于发端的客户侧芯片（host chip），收端设备 05 也称为位于收端的客户侧芯片，信道传输媒介 03 可以为光纤。客户侧芯片有时也称为为客户侧设备（host device）。其中，发端设备 01 与发端处理模块 02 之间可以通过连接单元接口（attachment unit interface，AUI）连接，收端设备 05 与收端处理模块 04 之间可以通过 AUI 连接。发端处理模块 02 和收端处理模块 04 可以为光模块（optical module）、电模块、连接器（connector）或其他在数据发送过程中对数据进行处理的模块。例如，该处理模块可以为 800LR 模块（800LR module，一种相干光模块）。并且，该通信系统中的发端设备 01、发端处理模块 02、信道传输媒介 03、收端处理模块 04 和收端设备 05 均可以支持双向传输，也可以支持单向传输，具体此处不做限定。

图 2 为图 1 所示通信系统中一种数据传输的过程示意图。如图 2 所示，在从发端设备 01 向收端设备 05 传输数据的过程中，发端设备 01 用于对该数据进行外码编码，然后向发端处理模块 02 传输经过外码编码的数据。发端处理模块 02 用于对经过外码编码的数据进行内码编码，得到经过外码编码和内码编码的数据，并将经过外码编码和内码编码的数据传输至信道传输媒介 03。信道传输媒介 03 用于将经过外码编码和内码编码的数据传输至收端处理模块 04。收端处理模块 04 用于对经过外码编码和内码编码的数据进行内码译码，并向收端设备 05 传输经过内码译码的数据。收端设备 05 用于对经过内码译码的数据进行外码译码。

应理解，内码中的“内”和外码中的“外”只是基于对数据进行操作的执行主体相对于信道传输媒介 03 的距离的远近来区分的。对内码进行操作的执行主体较靠近信道传输媒介，对外码进行操作的执行主体较远离信道传输媒介。在本申请实施例中，由于数据从发端设备 01 发出后经过发端处理模块 02 传输至信道传输媒介 03，然后从信道传输媒介 03 经过收端处理模块 04 传输至收端设备 05。经发端设备 01 编码的数据相对于经发端处理模块 02 编码的数据离信道传输媒介 03 较远，经收端设备 05 译码的数据相对于经收端处理模块 04 译码的数据离信道传输媒介 03 较远。因此经发端设备 01 编码的数据称为经过外码编码的数据，经发端处理模块 02 编码的数据称为经过内码编码的数据，经收端设备 05 译码的数据称为经过外码译码的数据，经收端处理模块 04 译码的数据称为经过内码译码的数据。在一种可能的实施方式中，上述的内码编码和外码编码都是采用 FEC 编码的方式，从而形成一种级联 FEC 的传输方案。例如，发端设备 01 可以采用 RS 码进行外码编码，发端处理模块 02 可以采用汉明（Hamming）码进行内码编码。又例如，发端设备 01 可以采用 RS 码进行外码编码，发端处理模块 02 可以采用博斯-查德胡里-霍昆格姆（Bose－Chaudhuri－Hocquenghem，BCH）码进行内码编码。

需要说明的是，以上内容是对本申请实施例提供的数据交织方法的应用场景的示例性说明，并不构成对于数据交织方法的应用场景的限定，本领域普通技术人员可知，随着业务需求的改变，其应用场景可以根据应用需求进行调整，本申请实施例对其不做一一列举。

对于上述采用级联FEC的传输方案，本申请设计一种包含“卷积交织”和“复用”的数据处理方案，以实现整体级联FEC方案性能较好且时延较低。使得该级联FEC传输方案能够应用于较多的传输场景，尤其适用于对需要较低传输时延的传输场景，如低时延数据中心互联场景。其中，数据处理是通过上述的发端处理模块02来实现的。

图3（a）为本申请实施例中发端处理模块的第一种数据处理示意图。如图3（a）所示，发端处理模块的物理媒体附加子层（Physical Medium Attachment，PMA）子层对来自多个同步的客户通道（client lane）的数据进行处理后，可以得到n条经过外码编码的物理编码子层（Physical Coding Sublayer，PCS）或FEC通道数据流，并进行标识锁定（alignment lock）和通道数据对齐处理得到对齐的n条通道数据流。然后根据对齐标识（Alignment marker）对n条通道的数据进行通道重排序（lane reorder）处理，使得n条通道的数据能够按照指定的顺序排列。经通道重排序的n条通道数据流送入所设计的包括卷积交织（convolutional interleaving）和复用（muxing）的处理器进行打乱数据顺序处理后送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括调制映射（mapping），信道交织（channelinterleaving），偏振划分（polarization distribution）或DSP成帧处理（framing）等。其中，这里n为大于1的正整数。

图3（b）为本申请实施例中发端处理模块的第二种数据处理示意图。如图3（b）所示，在有些实际应用场景中，经过通道数据对齐处理得到对齐的n条通道数据流已满足指定的顺序排列。此时不需要再进行通道重排序，直接将对齐的n条通道数据流送入所设计的包含卷积交织和复用的处理器进行交织打乱数据顺序处理后送入内码编码器进行内码编码。

应理解，在一些可能的实施方式中，区别于上述图3（a）和图3（b）所介绍的数据处理流程，经过通道数据对齐处理得到对齐的n条通道数据流也可以不进行卷积交织，直接进行复用处理，并送入内码编码器进行内码编码。

图3（c）为本申请实施例中发端处理模块的第三种数据处理示意图。如图3（c）所示，区别于上述图3（a）所示的数据处理流程，经通道重排序后的n条通道数据流不进行卷积交织，直接进行复用处理，并送入内码编码器进行内码编码。

图3（d）为本申请实施例中发端处理模块的第四种数据处理示意图。如图3（d）所示，区别于上述图3（b）所示的数据处理流程，经通道数据对齐处理后的n条通道数据流不进行卷积交织，直接进行复用处理，并送入内码编码器进行内码编码。

图3（e）为本申请实施例中通道数据对齐处理的示意图。应理解，上述“通道数据对齐处理”可以是采用现有标准定义的通道纠偏处理（lane de-skew），使得其输出的n条通道数据流的数据是完全对齐的。或者，上述“通道数据对齐处理”也可以仅是通道符号对齐，使得其输出的n条通道数据流上的数据是基于外码符号对齐，具体可以是基于一个外码符号对齐，也可以是基于多个外码符号对齐。图3（e）以两条通道数据流为例说明上述“通道数据对齐处理”具体操作，其假设外码是RS码，且一个RS码符号长度为10比特。其中，图3（e）中的场景(a)表明两条通道数据流存在偏差75个比特，AM0和AM1分别是通道数据流0和通道数据流1的对齐标识。图3(e)中的场景(b)是采用现有标准定义的通道纠偏处理（lane de-skew），使得其输出后的通道数据流0和通道数据流1不存在偏差。图3（e）中的场景

(c)是采用基于 1 个 RS 符号对齐处理，使得其输出后的通道数据流 0 的 1 个 RS 符号和通道数据流 1 的 1 个 RS 符号是对齐的，此时两个通道仍存在 70 比特的偏差。图 3（e）中的场景(d)是一种采用基于 2 个 RS 符号对齐处理，使得其输出后的通道数据流 0 的 2 个 RS 符号和通道数据流 1 的 2 个 RS 符号是对齐的，此时两个通道仍存在 60 比特的偏差。

图 3（f）为本申请实施例中发端处理模块的第五种数据处理示意图。如图 3（f）所示，发端处理模块的物理媒体附加子层（Physical Medium Attachment，PMA）子层对来自多个同步的客户通道（client lane）的数据进行处理后，可以得到 n 条经过外码编码的物理编码子层(Physical Coding Sublayer，PCS)或 FEC 通道数据流，并进行标识锁定(alignment lock)和通道数据对齐处理得到对齐的 n 条通道数据流。然后根据对齐标识（Alignment marker）对 n 条通道的数据进行通道重排序（lane reorder）处理，使得 n 条通道的数据能够按照指定的顺序排列。经通道重排序的 n 条通道数据流送入所设计的包含卷积交织（convolutional interleaving）和分组交织（block interleaving）的处理器进行交织打乱数据顺序处理后送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括调制映射（mapping），信道交织（channel interleaving），偏振划分（polarization distribution）或 DSP 成帧处理（framing）等。其中，这里 n 为大于 1 的正整数。

图 3（g）为本申请实施例中发端处理模块的第六种数据处理示意图。如图 3（g）所示，发端处理模块的物理媒体附加子层（Physical Medium Attachment，PMA）子层对来自多个同步的客户通道（client lane）的数据进行处理后，可以得到 n 条经过外码编码的物理编码子层（Physical Coding Sublayer，PCS）或 FEC 通道数据流，统称通道数据流，并进行标识锁定（alignment lock）和通道数据对齐处理得到对齐的 n 条通道数据流。然后根据对齐标识（Alignment marker）对 n 条通道的数据进行通道重排序（lane reorder）处理，使得 n 条通道的数据能够按照指定的顺序排列。经通道重排序的 n 条通道数据流送入所设计的包含第一分组交织、卷积交织（convolutional interleaving）和第二分组交织处理器进行交织打乱数据顺序处理后送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括调制映射（maPPing），信道交织（channel interleaving），偏振划分（Polarization distribution）或 DSP 成帧处理（Framing）等。其中，这里 n 为大于 1 的正整数。

图 3（h）为本申请实施例中发端处理模块的第七种数据处理示意图。如图 3（h）所示，发端处理模块的物理媒体附加子层（Physical Medium Attachment，PMA）子层对来自多个同步的客户通道（client lane）的数据进行处理后，可以得到 n 条经过外码编码的物理编码子层(Physical Coding Sublayer，PCS)或 FEC 通道数据流，并进行标识锁定(alignment lock)和通道数据对齐处理得到对齐的 n 条通道数据流。然后根据对齐标识（Alignment marker）对 n 条通道的数据进行通道重排序（lane reorder）处理，使得 n 条通道的数据能够按照指定的顺序排列。经通道重排序的 n 条通道数据流送入所设计的包含分组交织（block interleaving）和卷积交织（convolutional interleaving）的处理器进行交织打乱数据顺序处理后送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传

输媒介进行传输。该数据处理可包括调制映射(mapping)，信道交织(channel interleaving)，偏振划分（polarization distribution）或DSP成帧处理（framing）等。其中，这里n为大于1的正整数。

图3（i）为本申请实施例中发端处理模块的第八种数据处理示意图。如图3（i）所示，发端处理模块的物理媒体附加子层（Physical Medium Attachment，PMA）子层对来自n个同步的客户通道（client lane）比如AUI-n接口的数据进行处理后，可以得到n条经过外码编码通道数据流。此处PMA子层仅需要对来自每个客户通道的数据进行时钟数据恢复(clock data recovery，CDR)，PAM4符号解调制等信号恢复操作处理得到一条通道数据流，不需要进行AM锁定、通道纠偏以及通道重排序等其他复杂操作。n条通道数据流送入所设计的包含卷积交织（convolutional interleaving）的处理器进行交织打乱数据顺序处理后送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括调制映射（mapping），信道交织（channel interleaving），偏振划分（polarization distribution）或DSP成帧处理（framing）等。其中，这里n为大于1的正整数。

应理解，在有些实际应用场景中，RS外码编码是采用两个编码器编码后进行交织处理，比如采用双向交织（2-way interleavering），使得序号为偶数的通道数据流上的RS符号是按照“A B A B A B…”传输，而序号为奇数的通道数据流上的RS符号是按照“B A B A BA …”传输，其中A和B为不同编码器生成的两个RS符号。对于采用基于2个RS符号对齐处理，一种实现效果是使得同一时刻所有序号为偶数的通道数据流上的RS符号由相同编码器生成且所有序号为奇数的通道数据流上的RS符号由另一相同的编码器生成；另一种实现效果是同一时刻所有通道数据流上的RS符号均由相同编码器生成。这里不对具体方式进行限定。

图4（a）为本申请实施例中针对收端处理模块的第一种数据处理示意图。如图4（a）所示，收端处理模块接收来自信道传输媒介的数据流。当发端处理模块数据流是经过调制映射（mapping），信道交织(channel interleaving)，偏振划分(polarization distribution)或DSP成帧处理（framing）等数据处理时，收端处理模块先进行相应的数据逆处理后再送入内码译码器进行译码。内码译码后数据流送入卷积解交织和解复用处理器处理后得到n条通道数据流，再送入PMA子层。PMA子层将数据流进行处理后送入收端设备进行外码译码。这里收端处理模块中卷积解交织（convolutional de-interleaving）和解复用（de-muxing）处理是发端处理模块中卷积交织和复用处理的逆操作。其中卷积解交织是发端处理模块中卷积交织的逆操作，解复用是发端处理模块中复用的逆操作。下面对发端处理模块中的卷积交织和复用处理进行详细描述。收端处理模块的卷积解交织和解复用处理是如图3（a）和图3（b）所示发端处理模块中卷积交织和复用处理的逆操作，本领域普通技术人员可知，此处不再赘述。

图4（b）为本申请实施例中针对收端处理模块的第二种数据处理示意图。如图4（b）所示，收端处理模块接收来自信道传输媒介的数据流。当发端处理模块数据流是经过调制映射（mapping），信道交织(channel interleaving)，偏振划分(polarization distribution)或DSP成帧处理（framing）等数据处理时，收端处理模块先进行相应的数据逆处理后再送入

内码译码器进行译码。内码译码后数据流送入解分组交织和解卷积交织处理器处理后得到 n 条通道数据流，再送入 PMA 子层。PMA 子层将数据流进行处理后送入收端设备进行外码译码。这里收端处理模块中解分组交织和解卷积交织处理是如图 3（f）所示发端处理模块中分组交织和卷积交织处理的逆操作。其中解卷积交织是发端处理模块中卷积交织的逆操作，解分组交织是发端处理模块中分组交织的逆操作。

图 4（c）为本申请实施例中针对收端处理模块的第三种数据处理示意图。如图 4（c）所示，收端处理模块接收来自信道传输媒介的数据流。当发端处理模块数据流是经过调制映射（mapping），信道交织（channel interleaving），偏振划分（Polarization distribution）或 DSP 成帧处理（Framing）等数据处理时，收端处理模块先进行相应的数据逆处理后再送入内码译码器进行译码。内码译码后数据流送入解第二分组交织、解卷积交织和解第一分组交织处理后得到 n 条通道数据流，再送入 PMA 子层。PMA 子层将数据流进行处理后送入收端设备进行外码译码。这里收端处理模块中解第一分组交织、解卷积交织和解第二分组交织解处理分别是如图 3(g)所示发端处理模块中第一分组交织，卷积交织和第二分组交织的逆操作。下面对发端处理模块中第一分组交织、卷积交织和第二分组交织处理进行详细描述。收端处理模块的解第一分组交织，卷积解交织和解第二分组交织分别是发端处理模块中第一分组交织，卷积交织和第二分组交织的逆操作，本领域普通技术人员可知，此处不再赘述。

下面首先提供几种本申请实施例可应用的几种具体场景。需要说明的是，为了便于介绍，下面各具体场景都以“通道数据对齐处理”采用通道纠偏处理为例进行描述。

图 5 为发端设备采用 1×800G 接口的 32 条 PCS 通道数据流的示意图。如图 5 所示，发端设备将待传输的 1 路 800GbE 业务数据流进行 KP4 RS(544,514)码外码编码得到 32 条 PCS 通道（PCS lane）数据流。PCS 通道数据流 0-15 中每条数据流间隔 68 个符号总共 16*68=1088 个符号，其包含了 2 个 RS 码码字。每条 PCS 通道数据流中相邻 2 个符号来自不同 RS 码码字，且相邻两条 PCS 通道数据流的同个位置的 2 个符号来自不同 RS 码码字。类似的，PCS 通道数据流 16-31 中每条数据流间隔 68 个符号总共 16*68=1088 个符号，其包含了 2 个 RS 码码字。每条 PCS 通道数据流中相邻 2 个符号来自不同 RS 码码字，且相邻两条 PCS 通道数据流的同个位置的 2 个符号来自不同 RS 码码字。32 条 PCS 通道数据流经 PMA 处理后通过连接单元接口 800GAUI-8 送入发端处理模块。

基于上述图 3（a）所示的发端处理模块的数据处理示意图，发端处理模块利用 PCS 通道已知的对齐标识（Alignment marker）进行通道数据流的标识锁定（alignment lock）。这里 32 个通道的已知对齐标识各不相同（见《Ethernet Technology Consortium 800G Specification》）。发端处理模块接着对 32 个通道数据流进行通道纠偏处理（lane de-skew）得到对齐的 32 条通道数据流。然后根据对齐标识对 n=32 条通道的数据进行通道重排序（lane reorder）处理，使得 n=32 条通道的数据能够按照指定的顺序排列。一种顺序是跟图 5 一样通道数据流从上往下按从 0 到 31 排序。

图 6 为发端设备采用 2×400G 接口的 32 条 PCS 通道数据流的示意图。如图 6 所示，发端设备将待传输的 2 路 400GbE 业务数据流进行 KP4 RS(544,514)码外码编码得到 2 路总共 32 条 PCS 通道数据流，其中每路包括 16 条 PCS 通道数据流。PCS 通道数据流 0-15 或 PCS 通道

数据流 16-31 中每条数据流间隔 68 个符号总共 16*68=1088 个符号，其包含了 2 个 RS 码码字。每条 PCS 通道数据流中相邻 2 个符号来自不同 RS 码码字，且相邻两条 PCS 通道数据流的同个位置的 2 个符号来自不同 RS 码码字。32 条 PCS 通道数据流经 PMA 处理后通过连接单元接口 2×400GAUI-4 送入发端处理模块。

基于上述图 3（a）所示的发端处理模块的数据处理示意图，发端处理模块利用 PCS 通道 0-15 或 PCS 通道 16-31 已知的对齐标识（Alignment marker）进行 16 个通道数据流的标识锁定（alignment lock）。这里 PCS 通道 0-15 可认为是第 0 路 400G 中的 PCS 通道 0-15，PCS 通道 16-31 可认为是第 1 路 400G 中的 PCS 通道 0-15。第 0 路 400G 中的 16 个通道已知的对齐标识和第 1 路中的 16 个通道已知的对齐标识相同。发端处理模块接着对 32 个通道数据流进行通道纠偏处理（lane de-skew）得到对齐的 32 条通道数据流。然后根据 PCS 通道 0-15 或 PCS 通道 16-31 的对齐标识对 16 条通道的数据进行通道重排序（lane reorder）处理，使得 16 条通道的数据能够按照指定的顺序排列。最后使得 32 条通道的数据能够按照指定的顺序排列。一种顺序是跟图 6 一样通道数据流从上往下按从 0 到 31 排序。

图 7 为发端设备采用 4×200G 接口的 32 条 PCS 通道数据流的示意图。如图 7 所示，发端设备将待传输的 4 路 200GbE 业务数据流进行 KP4 RS(544,514)码外码编码得到 4 路总共 32 条 PCS 通道数据流，其中每路包括 8 条 PCS 通道数据流。PCS 通道数据流 0-7、PCS 通道数据流 8-15、PCS 通道数据流 16-23 或 PCS 通道数据流 24-31 中每条数据流间隔 136 个符号总共 8*136=1088 个符号，其包含了 2 个 RS 码码字。每条 PCS 通道数据流中相邻 2 个符号来自不同 RS 码码字，且相邻两条 PCS 通道数据流的同个位置的 2 个符号来自不同 RS 码码字。32 条 PCS 通道数据流经 PMA 处理后通过连接单元接口 4×200GAUI-2 送入发端处理模块。

基于上述图 3（a）所示的发端处理模块的数据处理示意图，发端处理模块利用 PCS 通道 0-7、PCS 通道 8-15、PCS 通道 16-23 或 PCS 通道 24-31 已知的对齐标识（Alignment marker）进行 8 个通道数据流的标识锁定（alignment lock）。这里 PCS 通道 0-7、PCS 通道 8-15、PCS 通道 16-23 或 PCS 通道 24-31 可分别认为是第 0 路、第 1 路、第 2 路或第 3 路 200G 中的 PCS 通道 0-7。发端处理模块接着对 32 个通道数据流进行通道纠偏处理（lane de-skew）得到对齐的 32 条通道数据流。然后根据 PCS 通道 0-7、PCS 通道 8-15、PCS 通道 16-23 或 PCS 通道 24-31 的对齐标识对 8 条通道的数据进行通道重排序（lane reorder）处理，使得 8 条通道的数据能够按照指定的顺序排列。最后使得 32 条通道的数据能够按照指定的顺序排列。一种顺序是跟图 7 一样通道数据流从上往下按从 0 到 31 排序。

图 8 为发端设备采用 8×100G 接口的 32 条 FEC 通道数据流的一种示意图。如图 8 所示，发端设备将待传输 8 路 100GbE 业务数据流进行 KP4 RS(544,514)码外码编码得到 8 路总共 32 条 FEC 通道(FEC lane)数据流，其中每路包括 4 条 FEC 通道数据流。当采用 2 个 KP4 RS(544,514)码字交织的“100G RS-FEC-Int”模式时，FEC 通道数据流 0-3、FEC 通道数据流 4-7、FEC 通道数据流 8-11、FEC 通道数据流 12-15、FEC 通道数据流 16-19、FEC 通道数据流 20-23、FEC 通道数据流 24-27 或 FEC 通道数据流 28-31 中每条数据流间隔 272 个符号总共 4*272=1088 个符号，其包含了 2 个 RS 码码字。每条 FEC 通道数据流中相邻 2 个符号来自不同 RS 码码字，且相邻两条 FEC 通道数据流的同个位置的 2 个符号来自不同 RS 码码字。32 条 FEC 通道数据

流经 PMA 处理后通过连接单元接口 8×100GAUI-1 送入发端处理模块。

图 9 为发端设备采用 8×100G 接口的 32 条 FEC 通道数据流的另一种示意图。如图 9 所示，区别于上述图 8 的场景，本场景中发端设备采用“100G RS-FEC”模式，FEC 通道数据流 0-3、FEC 通道数据流 4-7、FEC 通道数据流 8-11、FEC 通道数据流 12-15、FEC 通道数据流 16-19、FEC 通道数据流 20-23、FEC 通道数据流 24-27 或 FEC 通道数据流 28-31 中每条数据流间隔 136 个符号总共 4*136=544 个符号，其包含了 1 个 RS 码码字。32 条 FEC 通道数据流经 PMA 处理后通过连接单元接口 8×100GAUI-1 送入发端处理模块。

基于上述图 3（a）所示的发端处理模块的数据处理示意图，发端处理模块利用 FEC 通道 0-3、FEC 通道 4-7、FEC 通道 8-11、FEC 通道 12-15、FEC 通道 16-19、FEC 通道 20-23、FEC 通道 24-27 或 FEC 通道 28-31 已知的对齐标识（Alignment marker）进行 4 个通道数据流的标识锁定（alignment lock）。这里 FEC 通道 0-3、FEC 通道 4-7、FEC 通道 8-11、FEC 通道 12-15、FEC 通道 16-19、FEC 通道 20-23、FEC 通道 24-27 或 FEC 通道 28-31 可分别认为是第 0 路、第 1 路、第 2 路、第 3 路、第 4 路、第 5 路、第 6 路或第 7 路 100G 中的 FEC 通道 0-3。发端处理模块接着对 32 个通道数据流进行通道纠偏处理（lane de-skew）得到对齐的 32 条通道数据流。然后根据 FEC 通道 0-3、FEC 通道 4-7、FEC 通道 8-11、FEC 通道 12-15、FEC 通道 16-19、FEC 通道 20-23、FEC 通道 24-27 或 FEC 通道 28-31 的对齐标识对 4 条通道的数据进行通道重排序（lane reorder）处理，使得 4 条通道的数据能够按照指定的顺序排列。最后使得 32 条通道的数据能够按照指定的顺序排列。一种顺序是跟图 8 和图 9 一样通道数据流从上往下按从 0 到 31 排序。

图 10 为本申请实施例提供的数据处理方法的一种流程示意图。

1001、对 n 条通道数据流分别进行卷积交织得到 n 条第一数据流。

本实施例中，通道数据流可以是 PCS 通道数据流或 FEC 通道数据流，具体此处不做限定。n 条通道数据流都是经过第一 FEC 编码的数据流，也就是上文中介绍的经过外码编码的数据流，其中，n 为大于 1 的整数。例如，该外码编码可以采用 RS 码，经过外码编码后的 n 条数据流可以包括多个 RS 码字。在实际应用中也可以采用其他的编码方式进行外码编码，为了便于描述，下文统一用 RS 码字来表示外码编码后生成的码字。应理解，经过外码编码后的每 a 个码字分布在 b 条通道数据流中，其中，a≤b≤n，n 能被 b 整除，a 为大于或等于 1 的整数。在上述图 5-图 9 所示的不同应用场景中，a 和 b 的取值也可能不同。以图 5 所示的应用场景为例，n=32，a=2，b=16，即每 2 个码字分布在 16 条通道数据流中。在图 6-图 9 其他几个应用场景中 a 和 b 的取值可以通过附图推导得出，此处不再一一赘述。需要说明的是，本申请中外码码长取值是以符号为单位来统计的，其中，符号可以包括一个或多个比特。例如，外码为采用的 KP4 RS(544,514)码，码长 N=544 个符号，一个符号包含 10 个比特。

作为一个示例，当 a=1，表示在发端设备 01 处经过外码编码的码字不进行交织，直接分布在 b 条通道数据流中。如图 9 所示，a=1，发端设备 01 处经过外码 KP4 编码的 N=544 个符号的码字不进行交织，直接分布在 b=4 条通道数据流中，图 9 中所示的 1 个虚框中 544 个符号来自同一个 KP4 码字，且在每个虚框内一条通道数据流中连续的 N/b=544/4=136 个符号来

自同一个 KP4 码字。

作为又一个示例，当 a>1，表示在发端设备 01 处经过外码编码的 a 个码字先进行交织，再分布在 b 条通道数据流中。如图 8 所示，a=2，发端设备 01 处经过外码 KP4 编码的 2 个码字总共 aN=2*544=1088 个符号先进行双向符号交织（two-way symbol interleavering），再分布在 b=4 条通道数据流中，图 8 中所示的 1 个虚框中包含 1088 个符号来自 a=2 个 KP4 码字，在每个虚框内一条通道数据流中连续的 2N/b=2*544/4=272 个符号来自 a=2 个 KP4 码字且相邻两个符号来自不同的 KP4 码字。如图 5 所示，a=2，发端设备 01 处经过外码 KP4 编码的 2 个码字总共 aN=2*544=1088 个符号先进行双向符号交织，再分布在 b=16 条通道数据流中，图 5 中所示的 1 个虚框中包含 1088 个符号来自 a=2 个 KP4 码字，在每个虚框内一条通道数据流中连续的 2N/b=2*544/16=68 个符号来自 a=2 个 KP4 码字且相邻两个符号来自不同的 KP4 码字。

需要说明的是，经过卷积交织后每条第一数据流中连续的 z 个符号来自 z 个不同的码字，其中，z 为大于 1 的整数。下面结合卷积交织的具体实现方式进行介绍。

图 11 为本申请实施例中对 n 条通道数据流分别进行卷积交织的一种结构示意图。如图 11 所示，可以通过 n 个卷积交织器分别对 n 条通道数据流进行卷积交织，每条通道数据流经过卷积交织后可以得到数据顺序打乱的第一数据流。需要说明的是，在本实施例中，每个卷积交织器都采用类似的方式对输入的通道数据流进行卷积交织。具体地，每个卷积交织器包括 p 条延迟线，每个卷积交织器根据 p 条延迟线对输入的通道数据流进行延迟得到第一数据流。其中，p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q。每个存储单元用于存储 d 个符号，z=p*d。每条通道数据流中的符号按照 p 条延迟线的序号依次输入到 p 条延迟线，每条延迟线单次输入 d 个符号且单次输出 d 个符号，第一数据流中连续的 p*d 个符号包括每条延迟线输出的 d 个符号。其中，Q 为大于或等于 1 的整数，d 为大于或等于 1 的整数。例如，p 条延迟线分别包括 0 个存储单元、Q 个存储单元、2Q 个存储单元、…、(p-1)Q 个存储单元，每个存储单元用于存储 d 个符号，那么，p 条延迟线分别对应 p 个延迟取值，延迟取值包括 0 个符号、Q×d 个符号、2Q×d 个符号、…、(p-1)Q×d 个符号。需要说明的是，本申请中延迟取值是以符号为单位来统计的，其中，符号可以包括一个或多个比特。延迟线的延迟取值所包括的符号数量越多，就说明这条延迟线对数据流的延迟（也称为时延）越长。应理解，当延迟线没有包含存储单元时，延迟线的延迟为 0 个符号，即为无延迟透传。

下面结合附图对卷积交织器的具体结构进行介绍。

图 12（a）为本申请实施例中卷积交织器的第一种结构示意图。如图 12（a）所示，p 条延迟线中存储单元的数量按照 p 条延迟线的序号依次递减。即延迟线 0 具有(p-1)Q 个存储单元，每一条延迟线依次减少 Q 个存储单元，延迟线 p-1 具有 0 个存储单元。图 12（b）为本申请实施例中卷积交织器的第二种结构示意图。如图 12（b）所示，p 条延迟线中存储单元的数量按照 p 条延迟线的序号依次递增。即延迟线 0 具有 0 个存储单元，每一条延迟线依次增加 Q 个存储单元，延迟线 p-1 具有(p-1)Q 个存储单元。

需要说明的是，在同一个时刻，卷积交织器输入和输出的开关（switch）位于同一条延迟线上，待当前的延迟线单次输入 d 个符号并输出 d 符号后，再将开关切换到下一条延迟线

上，以保证每条通道数据流中的符号按照 p 条延迟线的序号依次输入到 p 条延迟线，且第一数据流中连续的 p*d 个符号包括每条延迟线输出的 d 个符号。其具体数据读出写入操作如下：从当前延迟线最靠近输出端口的存储单元读出 d 个符号。将当前延迟线中每个存储单元存储的 d 个符号转移至后一个存储单元。再将 d 个符号写入当前延迟线中最靠近输入端口的存储单元。之后，切换到下一条延迟线并重复上述操作，依次类推。在一种可能的实施方式中，若采用上述图 12(a)所示的卷积交织器，则卷积交织器的参数应满足$d(p*Q+1)\geq a*N/b$，N 为码字的长度，以使得卷积交织器输出的第一数据流中连续的 p*d 个符号来自 p*d 个不同的外码码字，其中，d≤a。在另一种可能的实施方式中，若采用上述图 12（b）所示的卷积交织器，则卷积交织器的参数应满足$d(p*Q-1)\geq a*N/b$，N 为码字的长度，以使得卷积交织器输出的第一数据流中连续的 p*d 个符号来自 p*d 个不同的外码码字，其中，d≤a。

应理解，当采用相同的参数 p、Q 和 d 时，图 12（a）的卷积交织处理和图 12（b）的卷积交织处理相互是对方的逆操作。也就是说，当发端处理模块采用图 12（a）所示的卷积交织结构，其收端处理模块对应的卷积解交织采用如图图 12（b）所示的结构。同理，当发端处理模块采用图 12（b）所示的卷积交织结构，其收端处理模块对应的卷积解交织采用如图 12（a）所示的结构。

还应理解，n 个卷积交织器中的任一个卷积交织器可以采用上述图 12（a）或图 12（b）中的一种。在实际应用中，n 个卷积交织器可以均采用图 12（a）所示的结构；或者，n 个卷积交织器也可以均采用图 12（b）所示的结构；又或者，也可以部分的卷积交织器采用图 12（a）所示的结构，剩余部分的卷积交织器采用图 12（b）所示的结构。

需要说明的是，在一些具体的应用场景中，以 n=32 为例，p 的取值可以为 2、3、4、6 或 8，d 的取值可以为 1 或 2。

为了便于介绍，下文中涉及卷积交织的实施例均以 n 个卷积交织器采用如图 12（a）所示结构为例进行说明，当然可以简单扩展到上述列举的其他结构，其具体实现方式本领域普通技术人员可知，此处不再赘述。

在一些可能的实施方式中，在对 n 条通道数据流分别进行卷积交织得到 n 条第一数据流之前，还可以对 n 路通道数据流进行通道重排序，以使得 n 路数据流按照预设顺序排列。以 32 条数据流为例，32 条数据流可以是按照 0 到 31 排序从上到下排列，当然可以简单扩展到其他顺序排列，其具体实现方式本领域普通技术人员可知，此处不再赘述。

在一些可能的实施方式中，在对 n 条通道数据流分别进行卷积交织得到 n 条第一数据流之前，还可以对 n 条通道数据流进行通道数据对齐处理。该通道数据对齐处理可以是采用现有标准定义的通道纠偏处理（lane de-skew），使得其输出的 n 条通道数据流的数据是完全对齐的。或者，上述“通道数据对齐处理”也可以仅是通道符号对齐，使得其输出的 n 条通道数据流上的数据是基于外码符号对齐，具体可以是基于一个外码符号对齐，也可以是基于多个外码符号对齐。关于通道数据对齐处理的详细介绍可以参考上述图 3（e）的相关描述，此处不再赘述。

1002、对 n 条第一数据流中的每 K 条第一数据流进行复用得到 1 条第二数据流，以得到共 m 条第二数据流。

图 13 为本申请实施例中对 n 条第一数据流进行复用的一种结构示意图。如图 13 所示，可以采用 m 个复用器进行复用处理。具体地，n 条第一数据流中的每 K 条第一数据流输入一个复用器，该复用器输出 1 条第二数据流。m 个复用器将输出共 m 条第二数据流，其中，m=n/K，k 为大于 1 的整数。为了描述方便，本申请实施例以整数 n 能被 K 整除为例进行描述。需要说明的是，n 条第一数据流包括 G 个第一数据流子集，G 为大于 1 的整数，不同第一数据流子集中的符号来自不同码字。在一种可能的实施方式中，若 K≤G，从任意 K 个第一数据流子集中分别选取一条第一数据流，即一个复用器输入的 K 条第一数据流分别来自 K 个第一数据流子集。在另一种可能的实施方式中，若 K＞G，则从每个第一数据流子集各取 K/G 条第一数据流，即一个复用器输入的 K 条第一数据流包括每个第一数据流子集中的 K/G 条第一数据流。例如，若 n=32、G=2、K=4、m=8，由于 K＞G，则需要分别从 2 个第一数据流子集各取 2 条第一数据流，以得到输入复用器的 4 条第一数据流。又例如，若 n=32、G=4、K=2、m=8，由于 K＜G，则需要从 4 个第一数据流子集任选 2 个第一数据流子集，并分别从这 2 个第一数据流子集各取 1 条第一数据流，以得到输入复用器的 2 条第一数据流。

需要说明的是，在一些具体的应用场景中，以 n=32 为例，K 的取值可以为 2、4 或 8。

应理解，上述第一数据流子集只是为了便于描述而引入的概念，在实际应用中，n 条第一数据流为一个整体并不存在划分，每个第一数据流子集可以视作 n 条第一数据流中的一条或多条数据流。

需要说明的是，由于参与复用的每条第一数据流中连续的 z 个符号来自 z 个不同的码字，因此，经过复用后每条第二数据流中连续的 y 个符号来自 y 个不同的码字，其中，y＞z。在一种可能的实施方式中，若 K≤G，则 y=K*z。在另一种可能的实施方式中，若 K＞G，则 y=G*z。

下面结合复用的具体实现方式进行介绍。为了便于描述，下文中统一将复用器输入的 K 条第一数据流记为复用输入数据流 0、复用输入数据流 1、复用输入数据流 2、...、和复用输入数据流 K-1。

图 14 为本申请实施例中复用器的第一种结构示意图。如图 14 所示，C_j^x表示复用输入数据流 j 中连续的Δ个符号，这Δ个符号来自Δ个不同的外码码字，其中，$0 \le j \le K-1$，若 K≤G，Δ为 z 的约数；若 K>G，$\Delta = z$。C_j^0，$C_j^1, \ldots, C_j^w$表示复用输入数据流 j 中连续的$\Delta \cdot W$个 RS 符号。需要说明的是，复用器输出的第二数据流包括多个第二数据流符号子集，每个第二数据流符号子集包括 K 组符号，每组符号包括Δ个符号。并且，每个第二数据流符号子集中相邻两组符号来自不同的第一数据流子集。具体地，第二数据流符号子集中第 j 组符号来自 K 条复用输入数据流中第 j 条复用输入数据流，其中，0≤j≤K-1。需要注意的是，若 K＞G，则 K 条复用输入数据流中相邻的 2 条复用输入数据流来自不同的第一数据流子集。更进一步需要注意的是，若 K＞G，每连续的 G 条复用输入数据流分别来自不同的第一数据流子集。

需要说明的是，由于每个第二数据流符号子集中相邻两组符号来自不同的第一数据流子集，所以经过复用后的第二数据流中连续的 y 个符号来自不同的 y 个码字，其中，y＞z（y=K*z 或 y=G*z）。应理解，如果只通过卷积交织想得到输出数据流中连续的 y 个符号来自不同的 y 个码字，所需要的延时较高。而本方案中，在减少了卷积交织所占用时长的基础上，结合复用处理仍能达到相当的性能。并且，进行复用处理所占用的时长较短，结合卷积交织和复用

就可以通过更短的延时实现相当的性能。

应理解，上述第二数据流符号子集只是为了便于描述而引入的概念，在实际应用中，第二数据流中的符号为一个整体并不存在划分，每个第二数据流符号子集可以视作第二数据流中的多个符号。

以图 14 为例，C_0^0，$C_1^0,\ldots,C_{K-1}^0$记为第 0 个第二数据流符号子集，C_0^1，$C_1^1,\ldots,C_{K-1}^1$记为第 1 个第二数据流符号子集,…,C_0^w，$C_1^w,\ldots,C_{K-1}^w$记为第 W 个第二数据流符号子集。以第 0 个第二数据流符号子集为例，C_0^0表示其中第 0 组的Δ个符号，C_1^0表示其中第 1 组的Δ个符号，…，C_{K-1}^0表示其中第 K-1 组的Δ个符号。可以看出，C_0^0来自复用输入数据流 0，C_1^0来自复用输入数据流 1，…，C_{K-1}^0来自复用输入数据流 K-1。应理解，假设Δ= z，C_0^0中连续的 z 个符号来自 z 个不同的码字，C_1^0中连续的 z 个符号来自 z 个不同的码字，…。如果要使得复用后第二数据流中连续的 y 个符号来自 y 个不同的码字，y＞z，则C_0^0和C_1^0要来自不同的第一数据流子集，即复用输入数据流 0 和复用输入数据流 1 来自不同的第一数据流子集。同理，C_1^0和C_2^0要来自不同的第一数据流子集，即复用输入数据流 1 和复用输入数据流 2 来自不同的第一数据流子集，依次类推。这样一来，复用后第二数据流中每连续的 2*z 个符号来自 2*z 个不同的外码码字。需要注意的是，若 K＞G，每连续的 G 条复用输入数据流来自不同的第一数据流子集，具体为复用输入数据流 0 至复用输入数据流 G-1 来自不同的第一数据流子集，复用输入数据流 G 与复用输入数据流 2*G-1 来自不同的第一数据流子集，依次类推。这样一来，复用后的第二数据中每连续 G*z 个符号来自 G*z 个不同的外码码字。

也就是说，通过上述方式，复用器将 K 条输入数据流中数据按照Δ个符号为单位以轮询方式输出到一条第二数据流中，即依次从复用输入数据流 0 至复用输入数据流 K-1 每条输出Δ个符号生成第二数据流，对应第二数据流数据顺序为C_0^0，$C_1^0,\ldots,C_{K-1}^0$，C_0^1，$C_1^1,\ldots,C_{K-1}^1,\cdots$，$C_0^w$，$C_1^w,\ldots,C_{K-1}^w,\cdots$。当 K≤G 时，从第一数据流子集中选取的 K 条第一数据流可以以任意顺序对应于复用器的复用输入数据流 0 至复用输入数据流 K-1，且Δ为 z的约数，复用输出后第二数据流中任意连续 K*z 个符号来自不同的外码码字。当 K＞G 时，从第一数据流子集选取的 K 条第一数据流需按照一定的规则对应于复用器的复用输入数据流 0 至复用输入数据流 K-1。具体的规则为复用器的每连续 G 条复用输入数据流来自不同的第一数据流子集。一种具体的方式为复用输入数据流 i*G 至复用输入数据流 i*G+G-1 分别来自第一数据流子集 0 至第一数据流子集 G-1，其中，$0 \le i < K/G$。这样可使得复用器输出的第二数据流中每连续的 G*z 个符号来自不同的外码码字。

1003、分别对 m 条第二数据流进行第二 FEC 编码得到编码数据流。

图 15 为本申请实施例中对 m 条第二数据流进行 FEC 编码的一种结构示意图。如图 15 所示，对 m 条第二数据流分别进行第二 FEC 编码，即上文提到的内码编码,内码编码的信息位长度小于或等于 y 个 RS 符号。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括调制映射（mapping），信道交织（channel interleaving），偏振划分（polarization distribution），或 DSP 成帧处理（framing）等。比如内码编码数据流可通过交织后再进行传输，以提升系统抗突发错误的能力。

本申请实施例中，n 路通道数据流都为经过外码编码后的码字流，在对 n 路数据流分别

进行卷积交织处理，并对卷积交织后的 n 路数据流进行数据流复用为 m 路第二数据流，随后进行内码编码。采用本申请提供的数据交织和复用处理方案，通过较短的时延就能实现复用处理后的 m 路数据流在连续输出的多个符号来自多个不同的外码码字，使得级联 FEC 方案在保证较好性能的基础上也有助于降低数据交织的时延。也就是说，本申请中卷积交织处理与数据复用相结合的方案使得级联 FEC 方案的整体时延较低，更适用于要求低时延的应用场景。

下面结合一些具体的实施例对上述图 10 所描述的数据处理方法的流程进行进一步介绍。

实施例 1：应用于 1×800G 接口的场景，内码编码信息位长度为 120 比特，采用 2:1、4:1、8:1 的复用器，采用通道纠偏处理。

基于图 3（a）所示的发端处理模块的数据处理示意图，发端处理模块利用 PCS 通道已知的对齐标识（Alignment marker）进行通道数据流的标识锁定（alignment lock）。这里 32 个通道的已知对齐标识各不相同（见《Ethernet Technology Consortium 800G Specification》）。发端处理模块接着对 32 个通道数据流进行通道纠偏处理（lane de-skew）得到完全对齐的 32 条通道数据流。然后根据对齐标识对 n=32 条通道的数据进行通道重排序（lane reorder）处理，使得 n=32 条通道的数据能够按照指定的顺序排列。一种顺序是跟图 5 一样通道数据流从上往下按从 0 到 31 排序。

经通道重排序的 n=32 条通道数据流送入所设计的卷积交织和复用处理器进行包括卷积交织和复用处理后，送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括调制映射（mapping），信道交织（channel interleaving），偏振划分（polarization distribution），或 DSP 成帧处理（framing）等。

在本实施例中，卷积交织处理采用如图 11 所示的结构，分别对 n=32 条 PCS 通道数据流进行卷积交织得到 n=32 条第一数据流。其中卷积交织器 0、卷积交织器 1、卷积交织器 2、…、卷积交织器 31 采用相同的交织结构。

图 16（a）为本申请实施例中卷积交织器的第三种结构示意图。如图 16（a）所示，其包含 p=3 条延迟线（delay line）。这 p=3 条延迟线分别包括 2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=2 个符号。也就是说，延迟线 0 的延迟取值为 4Q 个符号，延迟线 1 的延迟取值为 2Q 个符号，延迟线 2 的延迟取值为 0 个符号即无延迟。

如图 16(a)所示的$C_r(.)$表示该通道数据流$r(0 \le r \le n-1)$中一个 RS 符号。比如，$C_r(6t)$，$C_r(6t+1)$表示通道数据流r中当前输入给延迟线 0 的两个 RS 符号且$C_r(6t-12Q)$，$C_r(6t-12Q+1)$为延迟线 0 输出的两个 RS 符号；$C_r(6t+2)$，$C_r(6t+3)$表示通道数据流中接下来输入给延迟线 1 的两个 RS 符号且$C_r(6t-6Q+2)$，$C_r(6t-6Q+3)$为延迟线 1 输出的两个 RS 符号；$C_r(6t+4)$，$C_r(6t+5)$表示通道数据流中再接下来输入给延迟线 2 的两个 RS 符号且$C_r(6t+4)$，$C_r(6t+5)$为延迟线 2 输出的两个 RS 符号；$C_r(6t+6)$，$C_r(6t+7)$表示通道数据流中再随后输入给延迟线 0 的两个 RS 符号且$C_r(6t-12Q+6)$，$C_r(6t-12Q+7)$为延迟线 0 输出的两个 RS 符号，依次类推。结合图 5，可以看出当$6Q+2 \ge 68$即$Q \ge 11$时，卷积交织输出的$C_r(6t-12Q)$，$C_r(6t-12Q+1)$，$C_r(6t-6Q+2)$，$C_r(6t-6Q+3)$，$C_r(6t+4)$，$C_r(6t+5)$总共 6 个 RS 符号来自 6 个不同的 RS 码字。

图 16（b）为本申请实施例中卷积交织器的第四种结构示意图。如图 16（b）所示，在一种可能的实施方式中，选取$Q=11$，卷积交织器具体结构如图 16（b）所示。其对应的交织时延约为 22*2*3/2=66 个 RS 符号。采用如图 16（b）所示的卷积交织器分别对 32 条 PCS 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 5 所示的 PCS 通道数据流，不难理解，第一数据流 0-15 中的任一 RS 符号与第一数据流 16-31 中的任一 RS 符号来自不同的 RS 码字。因此 32 条第一数据流包含 G=2 个第一数据流子集，第一数据流 0-15 为第一数据流子集 0，第一数据流 16-31 为第一数据流子集 1。结合图 16（a），不难理解，第一数据流子集 0 中的任一个数据流r_0的 6 个输出符号$C_{r_0}(6t-12Q)$，$C_{r_0}(6t-12Q+1)$，$C_{r_0}(6t-6Q+2)$，$C_{r_0}(6t-6Q+3)$，$C_{r_0}(6t+4)$，$C_{r_0}(6t+5)$和第一数据流子集 1 中的任一个数据流r_1的 6 个输出符号$C_{r_1}(6t-12Q)$，$C_{r_1}(6t-12Q+1)$，$C_{r_1}(6t-6Q+2)$，$C_{r_1}(6t-6Q+3)$，$C_{r_1}(6t+4)$，$C_{r_1}(6t+5)$，总共 12 个 RS 符号来自 12 个不同的 RS 码字。

本实施例采用图 13 所示的复用处理的一种可能的实现为 G=2，K=2，m=16，产生 16 条第二数据流，其包含 16 个 2:1 复用处理模块，从第一数据流子集 0 和第一数据流子集 1 各自任取 1 条第一数据流作为 2:1 复用器的输入。

图 17（a）为本申请实施例中复用器的第二种结构示意图。如图 17（a）所示，2:1 复用器$i(0\le i\le 15)$的 2 条输入数据流分别为第一数据流i和第一数据流$i+16$。图中 C_j^x表示 2:1 复用器的复用输入数据流 j 中连续的$\Delta=6$个 RS 符号，其来自6个不同的外码 RS 码字。2:1 复用器将 2 条输入数据流中数据按照6个 RS 符号为单位以轮询方式输出到输出数据流中，即输出数据顺序为$C_0^0, C_1^0, C_0^1, C_1^1, \cdots, C_0^w, C_1^w \cdots$。需要说明的是由于 K=G=2，$\Delta=1,2,3$都也可以使得复用器输出数据流中的连续 12 个 RS 符号来自 12 个不同的 RS 码字。

本实施例采用图 13 所示的复用处理的另外一种可能的实现为 G=2，K=4，m=8，产生 8 条第二数据流，其包含 8 个 4:1 复用器。从第一数据流子集 0 和第一数据流子集 1 各自任取 2 条第一数据流作为 4:1 复用器的输入。

图 17（b）为本申请实施例中复用器的第三种结构示意图。如图 17（b）所示，4:1 复用器$i(0\le i\le 7)$的复用输入数据流 0，复用输入数据流 1，复用输入数据流 2，复用输入数据流 3 分别对应为第一数据流i，第一数据流$i+16$，第一数据流$i+8$，第一数据流$i+24$，即复用器的任意连续 2 条复用输入数据流来自不同的第一数据流子集。图中 C_j^x表示 4:1 复用器的复用输入数据流 j（$0\le j\le 3$）中连续的$\Delta=6$个 RS 符号，其来自6个不同的外码 RS 码字。4:1 复用器将 4 条输入数据流中数据按照6个 RS 符号为单位以轮询方式输出到输出数据流中，即输出数据顺序为$C_0^0, C_1^0, C_2^0, C_3^0, C_0^1, C_1^1, C_2^1, C_3^1, \ldots, C_0^w, C_1^w, C_2^w, C_3^w \cdots$。输出数据流中连续 12 个 RS 符号来自 12 个不同的 RS 码字。

本实施例采用图 13 所示的复用处理的另外一种可能的实现为 G=2，K=8，m=4，产生 4 条第二数据流，其包含 4 个 8:1 复用器。从第一数据流子集 0 和第一数据流子集 1 各自任取 4 条第一数据流作为 8:1 复用器的输入。

图 17（c）为本申请实施例中复用器的第四种结构示意图。如图 17（c）所示，8:1 复用器$i(0\le i\le 3)$的复用输入数据流 0 至 7 分别对应为第一数据流i，第一数据流$i+16$，第一数据流$i+8$，第一数据流$i+24$，第一数据流$i+4$，第一数据流$i+20$，第一数据流$i+12$，

第 i 数据流$i+28$，即复用器的任意连续 2 条复用输入数据流来自不同的第 i 数据流子集。需要说明的是，8:1 复用器i（$0 \le i \le 3$）的复用输入数据流 0 至 7 还可以分别对应为第 i 数据流i，第 i 数据流$i+16$，第 i 数据流$i+4$，第 i 数据流$i+20$，第 i 数据流$i+8$，第 i 数据流$i+24$，第 i 数据流$i+12$，第 i 数据流$i+28$，即复用器的任意连续 2 条复用输入数据流来自不同的第 i 数据流子集。图中 C_j^x 表示 8:1 复用器的复用输入数据流 j（$0 \le j \le 7$）中连续的$\Delta=6$个 RS 符号，其来自6个不同的外码 RS 码字。8:1 复用器将 8 条输入数据流中数据按照6个 RS 符号为单位以轮询方式输出到输出数据流中，即输出数据顺序为$C_0^0, C_1^0, \ldots., C_6^0, C_7^0, C_0^1, C_1^1, \cdots, C_6^1, C_7^1, \ldots, C_0^w$， $C_1^w, \cdots, C_6^w, C_7^w \cdots$。输出数据流中连续 12 个 RS 符号来自 12 个不同的 RS 码字。

上述 16 条、8 条或 4 条第二数据流分别进行内码编码，其中内码编码信息位长度为 120 比特。具体地，内码编码器分别将每条第二数据流中连续的 12 个 RS 符号总共 120 比特添加冗余，得到内码码字数据流。在一种可能的实施方式中，采用 Hamming(128,120)进行内码编码，分别将每条第二数据流中连续的 12 个 RS 符号总共 120 比特添加 8 比特冗余得到 128 比特的码字。在另一种可能的实施方式中，采用 BCH(136,120) 进行内码编码，分别将每条第二数据流中连续的 12 个 RS 符号总共 120 比特添加 16 比特冗余得到 136 比特的码字。

经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括调制映射（mapping），信道交织（channel interleaving），偏振划分（polarization distribution），或 DSP 成帧处理（framing）等。比如内码编码数据流可通过交织，以提升系统抗突发错误的能力。

采用本实施例 1 的数据交织编码方案，其 KP4 RS(544,514) + Hamming(128,120)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 4.5E-3，性能逼近该级联 FEC 方案的最优性能。

实施例 2：应用于 1×800G 接口的场景，内码编码信息位长度为 120 比特，采用 2:1、4:1、8:1 的复用器，采用通道符号对齐。

实施例 2 与实施例 1 的主要区别在于，实施例 2 是基于 2 个 RS 符号对齐得到对齐的 32 条通道数据流。

具体地，基于图 3（a）所示的发端处理模块的数据处理示意图，发端处理模块利用 PCS 通道已知的对齐标识（Alignment marker）进行通道数据流的标识锁定（alignment lock）。发端处理模块接着对 32 个通道数据流采用基于 2 个 RS 符号对齐得到对齐的 32 条通道数据流。然后根据对齐标识对 32 条通道的数据进行通道重排序（lane reorder）处理，使得 32 条通道的数据能够按照指定的顺序排列。一种排列顺序是跟图 3（a）一样通道数据流从上往下按从 0 到 31 排序。另一种排列顺序是使得“通道重排序”输出的 32 条通道中从上往下前 16 条通道数据流包含 PCS 通道数据流 0-15，后 16 条通道包含 PCS 通道数据流 16-31。应理解，此时前 16 条通道数据流中的具体顺序不做限定，后 16 条通道数据流中的具体顺序也不做限定。也就是说，图 3（a）中的通道数据流i不一定对应于 PCS 通道数据流i。

经通道重排序的 32 条通道数据流送入所设计的包含卷积交织和复用的处理器进行交织

打乱数据顺序处理后送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。应理解，本实施例采用的卷积交织和复用，以及内码编码方案均采用实施例 1 的方案。

采用本实施例 2 的数据交织编码方案，其 KP4 RS(544,514) + Hamming(128,120)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 4.5E-3，性能与实施例一的方案相当，且具有更低的整体时延。但是，和实施例 1 方案相比，本实施例 2 的方案在对抗系统突发错误较差。该方案适用于一些要求更低时延的场景。

实施例 3：应用于 1×800G 接口的场景，内码编码信息位长度为 160 比特，采用 2:1、4:1、8:1 的复用器，采用通道纠偏处理。

在实施例 1 基础上，本实施例考虑码长 160 比特的内码，对应地采用新设计的卷积交织器。

在本实施例中，卷积交织处理采用如图 11 所示的结构，分别对 n=32 条 PCS 通道数据流进行卷积交织得到 n=32 条第一数据流。其中卷积交织器 0、卷积交织器 1、卷积交织器 2、…、卷积交织器 31 采用相同的交织结构。

图 18（a）为本申请实施例中卷积交织器的第五种结构示意图。如图 18（a）所示，其包含 p=4 条延迟线（delay line）。这 4 条延迟线分别包括 3Q 个存储单元、2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=2 个符号。也就是，延迟线 0 的延迟取值为 6Q 个符号，延迟线 1 的延迟取值为 4Q 个符号，延迟线 2 的延迟取值为 2Q 个符号，延迟线 3 的延迟取值为 0 个符号即无延迟。

如图 18(a)所示的$C_r(.)$表示该通道数据流$r(0 \le r \le n-1)$中一个 RS 符号。比如，$C_r(8t)$，$C_r(8t+1)$表示通道数据流r中当前输入给延迟线 0 的两个 RS 符号且$C_r(8t-24Q)$，$C_r(8t-24Q+1)$为延迟线 0 输出的两个 RS 符号；$C_r(8t+2)$，$C_r(8t+3)$表示通道数据流中接下来输入给延迟线 1 的两个 RS 符号且$C_r(8t-16Q+2)$，$C_r(8t-16Q+3)$为延迟线 1 输出的两个 RS 符号；$C_r(8t+4)$，$C_r(8t+5)$表示通道数据流中再接下来输入给延迟线 2 的两个 RS 符号且$C_r(8t-8Q+4)$，$C_r(8t-8Q+5)$为延迟线 2 输出的两个 RS 符号；$C_r(8t+6)$，$C_r(8t+7)$表示通道数据流中再随后输入给延迟线 3 的两个 RS 符号且$C_r(8t+6)$，$C_r(8t+7)$为延迟线 3 输出的两个 RS 符号，依次类推。结合图 5，可以看出当$8Q+2 \ge 68$即$Q \ge 9$时，卷积交织输出的$C_r(8t-24Q)$，$C_r(8t-24Q+1$，$Cr(8t-16Q+2$，$Cr(8t-16Q+3$，$Cr(8t-8Q+4$，$Cr(8t-8Q+5$，$Cr(8t+6$，$Cr(8t+7$，总共 8 个 RS 符号来自 8 个不同的 RS 码字。

图 18（b）为本申请实施例中卷积交织器的第六种结构示意图。如图 18（b）所示，在一种可能的实施方式中，选取$Q=9$，卷积交织器具体结构如图 18（b）所示。其对应的交织时延约为 27*2*4/2=108 个 RS 符号。采用如图 18（b）所示的卷积交织器分别对 32 条 PCS 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 5 所示的 PCS 通道数据流，不难理解，第一数据流 0-15 中的任一 RS 符号与第一数据流 16-31 中的任一 RS 符号来自不同的 RS 码字。因此 32 条第一数据流包含 G=2 个第一数据流子集，第一数据流 0-15 为第一数据流子集 0，第一数据流 16-31 为第一数据流子集 1。结合图图 18（a），不难理解，第一数据流子集 0 的

任一个数据流r_0的 8 个输出符号$C_{r_0}(8t-24Q)$，$C_{r_0}(8t-24Q+1)$，$C_{r_0}(8t-16Q+2)$，$C_{r_0}(8t-16Q+3)$，$C_{r_0}(8t-8Q+4)$，$C_{r_0}(8t-8Q+5)$，$C_{r_0}(8t+6)$，$C_{r_0}(8t+7)$，和第一数据流子集 1 中的任一个数据流r_1的 8 个输出符号$C_{r_1}(8t-24Q)$，$C_{r_1}(8t-24Q+1)$，$C_{r_1}(8t-16Q+2)$，$C_{r_1}(8t-16Q+3)$，$C_{r_1}(8t-8Q+4)$，$C_{r_1}(8t-8Q+5)$，$C_{r_1}(8t+6)$，$C_{r_1}(8t+7)$，总共 16 个 RS 符号来自 16 个不同的 RS 码字。

本实施例采用图 13 所示的复用处理的一种可能的实现为 G=2,K=2，m=16，产生 16 条第二数据流，其包含 16 个 2:1 复用处理模块,从第一数据流子集 0 和第一数据流子集 1 各自任取 1 条第一数据流作为 2:1 复用器的输入。对应的 2:1 复用器一种具体实现方式如图 17（a）所示，2:1 复用器i（$0 \le i \le 15$）的 2 条输入数据流分别为第一数据流i和第一数据流$i+16$。图中 C_j^x表示 2:1 复用器的复用输入数据流 j（$0 \le j \le 1$）中连续的$\Delta=8$个 RS 符号，其来自8个不同的外码 RS 码字。2:1 复用器将 2 条输入数据流中数据按照8个 RS 符号为单位以轮询方式输出到输出数据流中，即输出数据顺序为$C_0^0, C_1^0, C_0^1, C_1^1, \cdots, C_0^w, C_1^w \cdots$。输出数据流中的连续 16 个 RS 符号来自 16 个不同的 RS 码字。需要说明由 K=G=2，$\Delta=1,2,4$也可以使得复用器输出数据流中的连续 16 个 RS 符号来自 16 个不同的 RS 码字。

本实施例采用图 13 所示的复用处理的另一种可能的实现为 G=2,K=4，m=8，产生 8 条第二数据流，其包含 8 个 4:1 复用器,从第一数据流子集 0 和第一数据流子集 1 各自任取 2 条第一数据流作为 4:1 复用器的输入。对应的 4:1 复用器一种具体实现方式如 17（b）所示，4:1 复用器i（$0 \le i \le 7$）的复用输入数据流 0，复用输入数据流 1，复用输入数据流 2，复用输入数据流 3 分别对应为第一数据流i，第一数据流$i+16$，第一数据流$i+8$，第一数据流$i+24$，即复用器的任意连续 2 条复用输入数据来自不同的第一数据符号子集。图中 C_j^x表示 4:1 复用器的复用输入数据流 j（$0 \le j \le 3$）中连续的$\Delta=8$个 RS 符号，其来自8个不同的外码 RS 码字。4:1 复用器将 4 条输入数据流中数据按照8个 RS 符号为单位以轮询方式输出到输出数据流中，即输出数据顺序为$C_0^0, C_1^0, C_2^0, C_3^0, C_0^1, C_1^1, C_2^1, C_3^1, \ldots, C_0^w, C_1^w, C_2^w, C_3^w \cdots$。输出数据流中连续 16 个 RS 符号来自 16 个不同的 RS 码字。

本实施例采用图 13 所示的复用处理的另外一种可能的实现为 G=2,K=8，m=4，生产 4 条第二数据流，其包含 4 个 8:1 复用器。从第一数据流子集 0 和第一数据流子集 1 各自任取 4 条第一数据流作为 8:1 复用器的输入，对应的 8:1 复用器一种具体实现方式如图 17（c）所示，8:1 复用器i（$0 \le i \le 3$）的复用输入数据流 0 至 7 分别对应为第一数据流i，第一数据流$i+16$，第一数据流$i+4$，第一数据流$i+20$，第一数据流$i+8$，第一数据流$i+24$，第一数据流$i+12$，第一数据流$i+28$。图中 C_j^x表示 8:1 复用器的复用输入数据流 j（$0 \le j \le 7$）中连续的$\Delta=8$个 RS 符号，其来自8个不同的外码 RS 码字。8:1 复用器将 8 条输入数据流中数据按照8个 RS 符号为单位以轮询方式输出到输出数据流中，即输出数据顺序为$C_0^0, C_1^0, C_2^0, C_3^0, C_4^0, C_5^0, C_6^0, C_7^0, C_0^1, C_1^1, C_2^1, C_3^1, C_4^1, C_5^1, C_6^1, C_7^1, \ldots, C_0^w, C_1^w, C_2^w, C_3^w \cdots$。输出数据流中连续 16 个 RS 符号来自 16 个不同的 RS 码字。

上述 16、8 或 4 条第二数据流分别进行内码编码，其中内码编码信息位长度为 160 比特。具体地，内码编码器分别将第二数据流中连续的 16 个 RS 符号总共 160 比特添加冗余，得到内码码字数据流。在一种可能的实施方式中，采用 Hamming(170,160)进行内码编码，分别将

每条第二数据流中连续的 16 个 RS 符号总共 160 比特添加 10 比特冗余得到 170 比特的码字。在另一种可能的实施方式中，采用 BCH(176,160) 进行内码编码，分别将每条第二数据流中连续的 16 个 RS 符号总共 160 比特添加 16 比特冗余得到 176 比特的码字。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。

采用本实施例的数据交织编码方案，当内码采用 Hamming(170,160)时，其 KP4 RS(544,514) + Hamming(170,160)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 4.3E-3，性能逼近该级联 FEC 方案的最优性能。当内码采用 BCH(176,160)时，其 KP4 RS(544,514) + BCH(176,160)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 8.3E-3，性能逼近该级联 FEC 方案的最优性能。

实施例 4：应用于 1×800G 接口的场景，内码编码信息位长度为 160 比特，采用 2:1、4:1、8:1 的复用器，采用通道符号对齐。

实施例 4 与实施例 3 的主要区别在于，实施例 4 是基于 2 个 RS 符号对齐得到对齐的 32 条通道数据流。

具体地，基于图 3（a）所示的发端处理模块的数据处理示意图，发端处理模块利用 PCS 通道已知的对齐标识（Alignment marker）进行通道数据流的标识锁定（alignment lock）。发端处理模块接着对 32 个通道数据流采用基于 2 个 RS 符号对齐得到对齐的 32 条通道数据流。然后根据对齐标识对 32 条通道的数据进行通道重排序（lane reorder）处理，使得 32 条通道的数据能够按照指定的顺序排列。一种排列顺序是跟图 3（a）一样通道数据流从上往下按从 0 到 31 排序。另一种排列顺序是使得“通道重排序”输出的 32 条通道中从上往下前 16 条通道数据流包含 PCS 通道数据流 0-15，后 16 条通道包含 PCS 通道数据流 16-31。应理解，此时前 16 条通道数据流中的具体顺序不做限定，后 16 条通道数据流中的具体顺序也不做限定。也就是，图 3（a）中的通道数据流i不一定对应于 PCS 通道数据流i。

经通道重排序的 32 条通道数据流送入所设计的包含卷积交织和复用的处理器进行交织打乱数据顺序处理后送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。应理解，本实施例采用的卷积交织和复用，以及内码编码方案均采用实施例 3 的方案。

采用本实施例 4 的数据交织编码方案，其 KP4 RS(544,514) + Hamming(160,120)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 4.5E-3，性能与实施例 3 的方案相当，且具有更低的整体时延。但是，和实施例 3 方案相比，本实施例 4 的方案在对抗系统突发错误较差。该方案适用于一些要求更低时延的场景。

采用本实施例 4 的数据交织编码方案，当内码采用 Hamming(170,160)时，其 KP4 RS(544,514) + Hamming(170,160)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 4.3E-3，性能逼近该级联 FEC 方案的最优性能。当内码采用 BCH(176,160)时，其 KP4 RS(544,514) + BCH(176,160)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 8.3E-3，性能逼近该级联 FEC 方案的最优性能。应理解，在采用和实施例 3 方案相同的内码方案下，本实施例 4 的方案性能与实施例 3 的一样，但在对抗

系统突发错误较差。该方案适用于一些要求更低时延的场景。

实施例 5：应用于 2×400G 接口的场景，内码编码信息位长度为 120 或 160 比特，采用 2:1、4:1、8:1 的复用器，采用通道纠偏处理。

区别于实施例 1-4，本实施例考虑客户侧接口为每个通道 100 Gb/s 的 2×400G，其接口细节见《IEEE Std 802.3ckTM/D3.0》。

具体地，基于上述图 3（a）所示的发端处理模块的数据处理示意图，发端处理模块利用 PCS 通道 0-15 或 PCS 通道 16-31 已知的对齐标识（Alignment marker）进行 16 个通道数据流的标识锁定（alignment lock）。发端处理模块接着对 32 个通道数据流进行通道纠偏处理（lane de-skew）得到对齐的 32 条通道数据流。然后根据 PCS 通道 0-15 或 PCS 通道 16-31 的对齐标识对 16 条通道的数据进行通道重排序（lane reorder）处理，使得 16 条通道的数据能够按照指定的顺序排列。最后使得 32 条通道的数据能够按照指定的顺序排列。一种顺序是跟图 6 一样通道数据流从上往下按从 0 到 31 排序。

经通道重排序的 32 条通道数据流送入所设计的包含卷积交织和复用的处理器进行交织打乱数据顺序处理后送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。

在一种可能的实施方式中，本实施例 5 中当采用的包含卷积交织和复用的处理器，以及内码编码跟实施例 1 中的方案一样时，该级联 FEC 方案的性能和时延与实施例 1 的一样。

在另一种可能的实施方式中，本实施例 5 中当采用的包含卷积交织和复用的处理器，以及内码编码跟实施例 3 中的方案一样时，该级联 FEC 方案的性能和时延与实施例 3 的一样。

实施例 6：应用于 2×400G 接口的场景，内码编码信息位长度为 120 或 160 比特，采用 2:1、4:1、8:1 的复用器，采用通道符号对齐，不进行通道重排序。

在实施例 5 方案基础上，实施例 6 给出了一种更低时延的实现方案。

具体地，基于上述图 3（b）所示的发端处理模块的数据处理示意图，发端处理模块利用 PCS 通道 0-15 或 PCS 通道 16-31 已知的对齐标识（Alignment marker）进行 16 个通道数据流的标识锁定（alignment lock）。发端处理模块接着对 32 个通道数据流进行采用基于 2 个 RS 符号对齐，得到对齐的 32 条通道数据流。然后直接将 32 条对齐的通道数据流送入所设计的包含卷积交织和复用的处理器进行交织打乱数据顺序处理后送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。

在一种可能的实施方式中，本实施例 6 中当采用的包含卷积交织和复用的处理器，以及内码编码跟实施例 2 中的方案一样时，该级联 FEC 方案的性能和时延与实施例 2 的一样。

在另一种可能的实施方式中，本实施例 6 中当采用的包含卷积交织和复用的处理器，以及内码编码跟实施例 4 中的方案一样时，该级联 FEC 方案的性能和时延与实施例 4 的一样。

实施例 7：应用于 4×200G 接口的场景，内码编码信息位长度为 120 或 160 比特，采用 4:1、8:1 的复用器，采用通道纠偏处理。

本实施例考虑客户侧接口为每个通道 100 Gb/s 的 4×200G，其接口细节见《IEEE Std 802.3ckTM/D3.0》。

基于图 3（a）所示的发端处理模块的数据处理示意图，发端处理模块利用 PCS 通道 0-7、PCS 通道 8-15、PCS 通道 16-23 或 PCS 通道 24-31 已知的对齐标识（Alignment marker）进行 8 个通道数据流的标识锁定（alignment lock）。这里 PCS 通道 0-7、PCS 通道 8-15、PCS 通道 16-23 或 PCS 通道 24-31 可分别认为是第 0 路、第 1 路、第 2 路或第 3 路 200G 中的 PCS 通道 0-7。发端处理模块接着对 32 个通道数据流进行通道纠偏处理（lane de-skew）得到对齐的 32 条通道数据流。然后根据 PCS 通道 0-7、PCS 通道 8-15、PCS 通道 16-23 或 PCS 通道 24-31 的对齐标识对 8 条通道的数据进行通道重排序（lane reorder）处理，使得 8 条通道的数据能够按照指定的顺序排列。最后使得 32 条通道的数据能够按照指定的顺序排列。一种顺序是跟图 7 一样通道数据流从上往下按从 0 到 31 排序。

经通道重排序的 32 条通道数据流送入所设计的包含卷积交织和复用的处理器进行交织打乱数据顺序处理后送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。

在本实施例中，卷积交织处理采用如图 11 所示的结构，分别对 n=32 条 PCS 通道数据流进行卷积交织得到 n=32 条第一数据流。其中卷积交织器 0、卷积交织器 1、卷积交织器 2、…、卷积交织器 31 采用相同的交织结构。

图 19（a）为本申请实施例中卷积交织器的第七种结构示意图。如图 19（a）所示，其包含 p=2 条延迟线（delay line）。这 2 条延迟线分别包括 Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=2 个符号。也就是，延迟线 0 的延迟取值为 2Q 个符号，延迟线 1 的延迟取值为 0 个符号即无延迟。

如图 19(a)所示的$C_r(\cdot)$表示该通道数据流$r(0 \le r \le n-1)$中一个 RS 符号。比如，$C_r(4t)$，$C_r(4t+1)$表示通道数据流中当前输入给延迟线 0 的两个 RS 符号且$C_r(4t-4Q)$，$C_r(4t-4Q+1)$为延迟线 0 输出的两个 RS 符号；$C_r(4t+2)$，$C_r(4t+3)$表示通道数据流中接下来输入给延迟线 1 的两个 RS 符号且$C_r(4t+2)$，$C_r(4t+3)$为延迟线 1 输出的两个 RS 符号；$C_r(4t+4)$，$C_r(4t+5)$表示通道数据流中再接下来输入给延迟线 0 的两个 RS 符号且$C_r(4t-4Q+4)$，$C_r(4t-4Q+5)$为延迟线 0 输出的两个 RS 符号，依次类推。结合图 7，可以看出当$4Q+2 \ge 136$即$Q \ge 34$时，卷积交织输出的连续 4 个 RS 符号$C_r(4t-4Q)$，$C_r(4t-4Q+1)$，$C_r(4t+2)$，$C_r(4t+3)$来自 4 个不同的 RS 码字。

图 19（b）为本申请实施例中卷积交织器的第八种结构示意图。如图 19（b）所示，在一种可能的实施方式中，选取$Q=34$，卷积交织器具体结构如图 19（b）所示。其对应的交织时延约为 34*2*2/2=68 个 RS 符号。采用如图 19（b）所示的卷积交织器分别对 32 条 PCS 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 7 所示的 PCS 通道数据流，不难理解，第一数据流 0-7 中的任一 RS 符号，第一数据流 8-15 中的任一 RS 符号，第一数据流 16-23 中的任一 RS 符号，与第一数据流 24-31 中的任一 RS 符号，均来自不同的 RS 码字。因此 32 条第一数据流包含 G=4 个第一数据流子集，第一数据流 0-7 为第一数据流子集 0，第一数据流 8-15 为第一数据流子集 1，第一数据流 16-23 为第一数据流子集 2，第一数据流 24-31 为

第一数据流子集 3。

本实施例采用如图 13 所示复用处理的一种可能实现方式为 G=4, K=4, m=8, 其包含 8 个 4:1 复用器，每个复用器将 4 条第一数据流进行复用得到 1 条第二数据流，总共生成 8 条第二数据流。从第一数据流子集 0 、第一数据流子集 1、第一数据流子集 2 和第一数据流子集 3 各自任取 1 条第一数据流作为 4:1 复用器的输入。

对应的 4:1 复用器一种具体实现方式如图 17（b）所示，4:1 复用器i（$0 \le i \le 7$）的复用输入数据流 0，复用输入数据流 1，复用输入数据流 2，复用输入数据流 3 分别对应为第一数据流i，第一数据流$i+16$，第一数据流$i+8$，第一数据流$i+24$。需要说明的是，4:1 复用器i（$0 \le i \le 7$）的复用输入数据流 0，复用输入数据流 1，复用输入数据流 2，复用输入数据流 3 还可以是分别对应为第一数据流i，第一数据流$i+8$，第一数据流$i+16$，第一数据流$i+24$。本实施例中，如图 17（b）所示的C_j^x表示 4:1 复用器的复用输入数据流 j（$0 \le j \le 3$）中连续的$\Delta=4$个 RS 符号，其来自4个不同的外码 RS 码字。4:1 复用器将 4 条输入数据流中数据按照4个 RS 符号为单位以轮询方式输出到输出数据流中，即输出数据顺序 $C_0^0, C_1^0, C_2^0 C_3^0, C_0^1, C_1^1, C_2^1, C_3^1, \ldots, C_0^w, C_1^w, C_2^w, C_3^w \cdots$。输出数据流中连续 16 个 RS 符号来自 16 个不同的 RS 码字。需要说明的是由于 K=G=4, 则从第一数据流子集 0 至 3 各取 1 条第一数据流可以以任意顺序分别对应于 4:1 复用器i（$0 \le i \le 7$）的复用输入数据流 0 至 3;$\Delta=1$,2仍可以使得第二数据流中连续的 16 个 RS 符号来自 16 个不同的 RS 码字。

本实施例采用如图 13 所示复用处理的另外一种可能实现方式为 G=4, K=8, m=4, 其包含 4 个 8:1 复用器，每个复用器将 8 条第一数据流进行复用得到 1 条第二数据流，总共生成 4 条第二数据流。从第一数据流子集 0、第一数据流子集 1、第一数据流子集 2 和第一数据流子集 3 各自任取 2 条第一数据流作为 8:1 复用器的输入。

对应的 8:1 复用器一种具体实现方式如图 17（c）所示，8:1 复用器i（$0 \le i \le 3$）的复用输入数据流 0 至 7 分别对应为第一数据流i，第一数据流$i+16$，第一数据流$i+8$，第一数据流$i+24$，第一数据流$i+4$，第一数据流$i+20$，第一数据流$i+12$，第一数据流$i+28$，其使得复用器的任意连续 Q=4 条复用输入数据流来自不同的第一数据流子集。需要说明的是，8:1 复用器i（$0 \le i \le 3$）的复用输入数据流 0 至 7 还可以分别对应为第一数据流i，第一数据流$i+8$，第一数据流$i+16$，第一数据流$i+24$，第一数据流$i+4$，第一数据流$i+12$，第一数据流$i+20$，第一数据流$i+28$。图中C_j^x表示 8:1 复用器的复用输入数据流 j（$0 \le j \le 7$）中连续的$\Delta=4$个 RS 符号，其来自4个不同的外码 RS 码字。8:1 复用器将 8 条输入数据流中数据按照4个RS符号为单位以轮询方式输出到输出数据流中，即输出数据顺序为C_0^0, C_1^0, ….,C_6^0, C_7^0, C_0^1, C_1^1, …,C_6^1, C_7^1,…,C_0^w, C_1^w,…, C_6^w, C_7^w…。输出数据流中连续 16 个 RS 符号来自 16 个不同的 RS 码字。

上述 8 或者 4 条第二数据流分别进行内码编码，其内码编码方案可以采用实施例 1 给出的编码方案，得到与实施例 1 相当的性能；也可以采用实施例 3 给出的编码方案，得到与实施例 3 相当的性能，此处不再赘述。

实施例 8：应用于 4×200G 接口的场景，内码编码信息位长度为 120 比特，采用 2:1 的

复用器，采用通道纠偏处理。

在实施例 7 基础上，本实施例考虑采用 2:1 复用器，对应地采用新设计的卷积交织器。

具体地，基于图 3（a）所示的发端处理模块的数据处理示意图，发端处理模块利用 PCS 通道 0-7、PCS 通道 8-15、PCS 通道 16-23 或 PCS 通道 24-31 已知的对齐标识（Alignment marker）进行 8 个通道数据流的标识锁定（alignment lock）。这里 PCS 通道 0-7、PCS 通道 8-15、PCS 通道 16-23 或 PCS 通道 24-31 可分别认为是第 0 路、第 1 路、第 2 路或第 3 路 200G 中的 PCS 通道 0-7。发端处理模块接着对 32 个通道数据流进行通道纠偏处理（lane de-skew）得到对齐的 32 条通道数据流。然后根据 PCS 通道 0-7、PCS 通道 8-15、PCS 通道 16-23 或 PCS 通道 24-31 的对齐标识对 8 条通道的数据进行通道重排序（lane reorder）处理，使得 8 条通道的数据能够按照指定的顺序排列。最后使得 32 条通道的数据能够按照指定的顺序排列。一种顺序是跟图 7 一样通道数据流从上往下按从 0 到 31 排序。

经通道重排序的 32 条通道数据流送入所设计的包含卷积交织和复用的处理器进行交织打乱数据顺序处理后送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。

在本实施例中，卷积交织处理采用如图 11 所示的结构，分别对 n=32 条 PCS 通道数据流进行卷积交织得到 n=32 条第一数据流。其中卷积交织器 0、卷积交织器 1、卷积交织器 2、…、卷积交织器 31 采用相同的交织结构。图 16（a）给出了一种卷积交织器结构，其包含 p=3 条延迟线（delay line）。这 3 条延迟线分别包括 2Q 个存储单元、Q 个存储单元，0 个存储单元，每个存储单元用于存储 d=2 个符号。也就是，延迟线 0 的延迟取值为 4Q 个符号，延迟线 1 的延迟取值为 2Q 个符号，延迟线 2 的延迟取值为 0 个符号即无延迟。

如图 16（a）所示所示的$C_r(\cdot)$表示该通道数据流r（$0 \le r \le n-1$）中一个 RS 符号。比如$C_r(6t)$，$C_r(6t+1)$表示通道数据流中当前输入给延迟线 0 的两个 RS 符号且$C_r(6t-12Q)$，$C_r(6t-12Q+1)$为延迟线 0 输出的两个 RS 符号；$C_r(6t+2)$，$C_r(6t+3)$表示通道数据流中接下来输入给延迟线 1 的两个 RS 符号且$C_r(6t-6Q+2)$，$C_r(6t-6Q+3)$为延迟线 1 输出的两个 RS 符号；$C_r(6t+4)$，$C_r(6t+5)$表示通道数据流中再接下来输入给延迟线 2 的两个 RS 符号且$C_r(6t+4)$，$C_r(6t+5)$为延迟线 2 输出的两个 RS 符号；$C_r(6t+6)$，$C_r(6t+7)$表示通道数据流中再随后输入给延迟线 0 的两个 RS 符号且$C_r(6t-12Q+6)$，$C_r(6t-12Q+7)$为延迟线 0 输出的两个 RS 符号，依次类推。结合图 7，可以看出当$6Q+2 \ge 136$即$Q \ge 23$时，卷积交织输出的$C_r(6t-12Q)$，$C_r(6t-12Q+1)$，$C_r(6t-6Q+2)$，$C_r(6t-6Q+3)$，$C_r(6t+4)$，$C_r(6t+5)$总共 6 个 RS 符号来自 6 个不同的 RS 码字。

图 20 为本申请实施例中卷积交织器的第九种结构示意图。如图 20 所示，在一种可能的实施方式中，选取$Q=23$，卷积交织器具体结构如图 20 所示。其对应的交织时延约为 46*2*3/2=138 个 RS 符号。采用如图 20 所示的卷积交织器分别对 32 条 PCS 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 7 所示的 PCS 通道数据流，不难理解，第一数据流 0-7 中的任一 RS 符号，第一数据流 8-15 中的任一 RS 符号，第一数据流 16-23 中的任一 RS 符号，与第一数据流 24-31 中的任一 RS 符号，均来自不同的 RS 码字。因此 32 条第一数据流包含 G=4 个第一数据流子集，第一数据流 0-7 为第一数据流子集 0，第一数据流 8-15 为

第一数据流子集 1，第一数据流 16-23 为第一数据流子集 2，第一数据流 24-31 为第一数据流子集 3。

本实施例采用如图 13 所示复用处理的一种可能实现方式为 G=4,K=2,m=16，其包含 16 个 2:1 复用器，每个复用器将 2 条第一数据流进行复用得到 1 条第二数据流，总共生成 16 条第二数据流。从第一数据流子集 0 、第一数据流子集 1、第一数据流子集 2 和第一数据流子集 3 中任意 2 个第一数据流子集各自任取 1 条第一数据流作为 2:1 复用器的输入，对应的 2:1 复用器一种具体实现方式如图 17（a）所示，2:1 复用器i（$0 \le i \le 15$）的复用输入数据流 0 和输入数据流 1 分别对应为第一数据流i，第一数据流$i+16$。图中 C_j^x表示 2:1 复用器的复用输入数据流 j（$0 \le j \le 1$）中连续的$\Delta=6$个 RS 符号，其来自6个不同的外码 RS 码字。2:1 复用器将 2 条输入数据流中数据按照6个 RS 符号为单位以轮询方式输出到输出数据流中，即输出数据顺序为$C_0^0, C_1^0, C_0^1, C_1^1, \ldots, C_0^w, C_1^w \cdots$。输出数据流中连续 12 个 RS 符号来自 12 个不同的 RS 码。需要说明的是由于 K<G,则$\Delta=1,2,3$仍可以使得第二数据流中连续 12 个 RS 符号来自 12 个不同的 RS 码。

上述 16 条第二数据流分别进行内码编码，其内码编码方案可以采用实施例 1 给出的编码方案，得到与实施例 1 相当的性能，此处不再赘述。

实施例 9：应用于 4×200G 接口的场景，内码编码信息位长度为 160 比特，采用 2:1 的复用器，采用通道纠偏处理。

在实施例 8 基础上，本实施例考虑码长 160 比特的内码，对应地采用新设计的卷积交织器。

具体地，本实施例采用图 18（a）给出的卷积交织器结构，其包含 p=4 条延迟线（delay line）。这 4 条延迟线分别包括 3Q 个存储单元、2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=2 个符号。也就是，延迟线 0 的延迟取值为 6Q 个符号，延迟线 1 的延迟取值为 4Q 个符号，延迟线 2 的延迟取值为 2Q 个符号，延迟线 3 的延迟取值为 0 个符号即无延迟。

如图18(a)所示的$C_r(\cdot)$表示该通道数据流$r(0 \le r \le n-1)$中一个RS符号。比如，$C_r(8t)$，$C_r(8t+1)$表示通道数据流中当前输入给延迟线0的两个RS符号且$C_r(8t-24Q), C_r(8t-24Q+1)$为延迟线 0 输出的两个 RS 符号；$C_r(8t+2)$，$C_r(8t+3)$表示通道数据流中接下来输入给延迟线 1 的两个 RS 符号且$C_r(8t-16Q+2)$，$C_r(8t-16Q+3)$为延迟线 1 输出的两个 RS 符号；$C_r(8t+4)$，$C_r(8t+5)$表示通道数据流中再接下来输入给延迟线 2 的两个 RS 符号且$C_r(8t-8Q+4)$，$C_r(8t-8Q+5)$为延迟线 2 输出的两个 RS 符号；$C_r(8t+6)$，$C_r(8t+7)$表示通道数据流中再随后输入给延迟线 3 的两个 RS 符号且$C_r(8t+6), C_r(8t+7)$为延迟线 3 输出的两个 RS 符号，$C_r(8t+8)$，$C_r(8t+9)$表示通道数据流中再随后输入给延迟线 0 的两个 RS 符号且$C_r(8t-24Q+8)$，$C_r(8t-24Q+9)$为延迟线 0 输出的两个 RS 符号，依次类推。结合图 7，可以看出当$8Q+2 \ge 136$即$Q \ge 17$时，卷积交织输出的$C_r(8t-24Q)$，$C_r(8t-24Q+1)$，$C_r(8t-16Q+2)$，$C_r(8t-16Q+3)$，$C_r(8t-8Q+4)$，$C_r(8t-8Q+5)$，$C_r(8t+6)$，$C_r(8t+7)$总共 8 个 RS 符号来自 8 个不同的 RS 码字。

图 21 为本申请实施例中卷积交织器的第十种结构示意图。如图 21 所示，在一种可能的

实施方式中，选取$Q=17$，卷积交织器具体结构如图 21 所示。其对应的交织时延约为 51*2*4/2=204 个 RS 符号。

采用如图 21 所示的卷积交织器分别对 32 条 PCS 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 7 所示的 PCS 通道数据流，不难理解，第一数据流 0-7 中的任一 RS 符号，第一数据流 8-15 中的任一 RS 符号，第一数据流 16-23 中的任一 RS 符号，与第一数据流 24-31 中的任一 RS 符号，均来自不同的 RS 码字。因此 32 条第一数据流包含 G=4 个第一数据流子集，第一数据流 0-7 为第一数据流子集 0，第一数据流 8-15 为第一数据流子集 1，第一数据流 16-23 为第一数据流子集 2，第一数据流 24-31 为第一数据流子集 3。

本实施例采用如图 13 所示复用处理的一种可能实现方式为 G=4, K=2, m=16，其包含 16 个 2:1 复用器，每个复用器将 2 条第一数据流进行复用得到 1 条第二数据流，总共生成 16 条第二数据流。从第一数据流子集 0、第一数据流子集 1、第一数据流子集 2 和第一数据流子集 3 中任意 2 个第一数据流子集各自任取 1 条第一数据流作为 2:1 复用器的输入。对应的 2:1 复用器一种具体实现方式如图 17（a）所示，2:1 复用器i（$0\le i\le 15$）的复用输入数据流 0 和复用输入数据流 1 分别对应为第一数据流i，第一数据流$i+16$。图中 C_j^x表示 2:1 复用器的复用输入数据流 j（$0\le j\le 1$）中连续的$\Delta=8$个 RS 符号，其来自8个不同的外码 RS 码字。2:1 复用器将 2 条输入数据流中数据按照8个 RS 符号为单位以轮询方式输出到输出数据流中，即输出数据顺序为$C_0^0, C_1^0, C_0^1, C_1^1, \ldots, C_0^w, C_1^w \cdots$。输出数据流中连续 16 个 RS 符号来自 16 个不同的 RS 码字。需要说明的是由于 K<G, 则$\Delta=1,2,4$仍可以使得第二数据流中连续 16 个 RS 符号来自 16 个不同的 RS 码。

上述 16 条第二数据流分别进行内码编码，其内码编码方案可以采用实施例 3 给出的编码方案，得到与实施例 3 相当的性能，此处不再赘述。

实施例 10：应用于 4×200G 接口的场景，内码编码信息位长度为 160 比特，采用通道符号对齐。

在实施例 7-9 中任意实施例的基础上，本实施例给出了一种更低时延的实现方案。

基于图 3(d)所示的发端处理模块的数据处理示意图，发端处理模块利用 PCS 通道 0-7、PCS 通道 8-15、PCS 通道 16-23 或 PCS 通道 24-31 已知的对齐标识（Alignment marker）进行 8 个通道数据流的标识锁定（alignment lock）。这里 PCS 通道 0-7、PCS 通道 8-15、PCS 通道 16-23 或 PCS 通道 24-31 可分别认为是第 0 路、第 1 路、第 2 路或第 3 路 200G 中的 PCS 通道 0-7。发端处理模块接着对 32 个通道数据流采用基于 2 个 RS 符号对齐得到对齐的 32 条通道数据流。然后直接将 32 条对齐的通道数据流送入所设计的包含复用的处理器进行处理后送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。

在一种可能的实施方式中，当本实施例复用处理以及内码编码方案均采用实施例 7 的方案时，其级联码在 AWGN 下性能与实施例 7 的方案相当，且具有更低的整体时延。但是，和实施例 7 方案相比，本实施例的方案在对抗系统突发错误较差。该方案适用于一些要求更低时延的场景。

在另一种可能的实施方式中，当本实施例的复用处理以及内码编码方案均采用实施例 8 的方案时，其级联码在 AWGN 下性能与实施例 8 的方案相当，且具有更低的整体时延。但是，和实施例 8 相比，本实施例的方案在对抗系统突发错误较差。该方案适用于一些要求更低时延的场景。

在又一种可能的实施方式中，当本实施例的复用处理以及内码编码方案均采用实施例 9 的方案时，其级联码在 AWGN 下性能与实施例 9 的方案相当，且具有更低的整体时延。但是，和实施例 9 相比，本实施例的方案在对抗系统突发错误较差。该方案适用于一些要求更低时延的场景。

实施例 11：应用于 8×100G 接口的场景，内码编码信息位长度为 120 比特或 160 比特，采用 8:1 的复用器，采用通道纠偏处理。

本实施例考虑客户侧接口为每个通道 100 Gb/s 的 8×100G 且采用“100G RS-FEC-Int”模式，其接口细节见《IEEE Std 802.3ckTM/D3.0》。

基于图 3（c）所示的发端处理模块的数据处理示意图，发端处理模块利用 FEC 通道 0-3、FEC 通道 4-7、FEC 通道 8-11、FEC 通道 12-15、FEC 通道 16-19、FEC 通道 20-23、FEC 通道 24-27 或 FEC 通道 28-31 已知的对齐标识（Alignment marker）进行 4 个通道数据流的标识锁定（alignment lock）。这里 FEC 通道 0-3、FEC 通道 4-7、FEC 通道 8-11、FEC 通道 12-15、FEC 通道 16-19、FEC 通道 20-23、FEC 通道 24-27 或 FEC 通道 28-31 可分别认为是第 0 路、第 1 路、第 2 路、第 3 路、第 4 路、第 5 路、第 6 路或第 7 路 100G 中的 FEC 通道 0-3。发端处理模块接着对 32 个通道数据流进行通道纠偏处理（lane de-skew）得到对齐的 32 条通道数据流。然后根据 FEC 通道 0-3、FEC 通道 4-7、FEC 通道 8-11、FEC 通道 12-15、FEC 通道 16-19、FEC 通道 20-23、FEC 通道 24-27 或 FEC 通道 28-31 的对齐标识对 4 条通道的数据进行通道重排序（lane reorder）处理，使得 4 条通道的数据能够按照指定的顺序排列。最后使得 32 条通道的数据能够按照指定的顺序排列。一种顺序是跟图 11 一样通道数据流从上往下按从 0 到 31 排序。经通道重排序的 32 条通道数据流不进行卷积交织，直接进行复用处理得到总共 16 条第二数据流，并送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。

参考图 8 所示的 PCS 通道数据流，不难理解，通道数据流 0-3 中的任一个 RS 符号，通道数据流 4-7 中的任一个 RS 符号，通道数据流 8-11 中的任一个 RS 符号，通道数据流 12-15 中的任一个 RS 符号，通道数据流 16-19 中的任一个 RS 符号，通道数据流 20-23 中的任一个 RS 符号，通道数据流 24-27 中的任一个 RS 符号，与通道数据流 28-31 中的任一个 RS 符号，均来自不同的 RS 码字，由于通道数据流不进行卷积交织，因此通道数据流 0-31 等同于第一数据流 0-31。因此 32 条第一数据流包含 G=8 个第一数据流子集，第一数据流 0-3 为第一数据流子集 0，第一数据流 4-7 为第一数据流子集 1，第一数据流 8-11 为第一数据流子集 2，第一数据流 12-15 为第一数据流子集 3，第一数据流 16-19 为第一数据流子集 4，第一数据流 20-23 为第一数据流子集 5，第一数据流 24-27 为第一数据流子集 6，第一数据流 28-31 为第一数据流子集 7。

本实施例采用如图 13 所示复用处理的一种可能实现方式为 G=8,K=8,m=4，其将 32 条通道数据流进行复用处理得到 4 条第二数据流。其中，第一数据流子集 0 中的某一条，第一数据流子集 1 中的某一条，第一数据流子集 2 中的某一条，第一数据流子集 3 中的某一条，第一数据流子集 4 中的某一条，第一数据流子集 5 中的某一条，第一数据流子集 6 中的某一条，和第一数据流子集 7 中的某一条，总共 8 条通道数据流作为 8:1 复用器i（$0 \le i \le 3$）的 8 条输入数据流。一种具体实现方式如图 17（c）所示，复用器i（$0 \le i \le 3$）将第一数据流i，第一数据流$i+16$，第一数据流$i+8$，第一数据流$i+24$，第一数据流$i+4$，第一数据流$i+20$，第一数据流$i+12$，第一数据流$i+28$分别对应于 8:1 复用器的复用输入数据流 0 至 7。图中C_j^x表示 8:1 复用器的复用输入数据流 j（$0 \le j \le 7$）中连续的$\Delta=2$个 RS 符号，其来自2个不同的外码 RS 码字。8:1 复用器将 8 条输入数据流中数据按照2个 RS 符号为单位以轮询方式输出到输出数据流中，即输出数据顺序为$C_0^0, C_1^0, \ldots, C_6^0, C_7^0, C_0^1, C_1^1, \cdots, C_6^1, C_7^1, \ldots, C_0^w, C_1^w, \cdots, C_6^w, C_7^w \cdots$。输出数据流中连续 16 个 RS 符号来自 16 个不同的 RS 码字。需要说明的是由于$K \le G$,则$\Delta=1$仍可以使得第二数据流中连续 16 个 RS 符号来自 16 个不同的 RS 码。

图 22 为本申请实施例中复用器的第五种结构示意图。如图 22 所示，复用器i($0 \le i \le 3$)将第一数据流i，第一数据流$i+4$，第一数据流$i+8$，第一数据流$i+12$，第一数据流$i+16$，第一数据流$i+20$，第一数据流$i+24$，第一数据流$i+28$依次分别对应于 8:1 复用器的复用输入数据流 0~7。图中 C_j^x表示 8:1 复用器的复用输入数据流 j（$0 \le j \le 7$）中连续的$\Delta=2$个 RS 符号，其来自2个不同的外码 RS 码字。8:1 复用器将 8 条输入数据流中数据按照2个 RS 符号为单位以轮询方式输出到输出数据流中，即输出数据顺序为$C_0^0, C_1^0, \ldots, C_6^0, C_7^0, C_0^1, C_1^1, \cdots, C_6^1, C_7^1, \ldots, C_0^w, C_1^w, \cdots, C_6^w, C_7^w \cdots$。输出数据流中连续 16 个 RS 符号来自 16 个不同的 RS 码字。需要说明的是由于$K \le G$,则$\Delta=1$仍可以使得第二数据流中连续 16 个 RS 符号来自 16 个不同的 RS 码。

上述 4 条第二数据流分别进行内码编码，其内码编码方案可以采用实施例 1 给出的编码方案，得到与实施例 1 相当的性能；也可以采用实施例 3 给出的编码方案，得到与实施例 3 相当的性能，此处不再赘述。

实施例 12：应用于 8×100G 接口的场景，内码编码信息位长度为 120 比特或 160 比特，采用 4:1 的复用器，采用通道纠偏处理。

在实施例 11 方案基础上，本实施例给出了复用处理采用 4:1 复用器时一种次低时延的实现方案。

基于图 3（a）所示的发端处理模块的数据处理示意图，发端处理模块利用 FEC 通道 0-3、FEC 通道 4-7、FEC 通道 8-11、FEC 通道 12-15、FEC 通道 16-19、FEC 通道 20-23、FEC 通道 24-27 或 FEC 通道 28-31 已知的对齐标识（Alignment marker）进行 4 个通道数据流的标识锁定（alignment lock）。这里 FEC 通道 0-3、FEC 通道 4-7、FEC 通道 8-11、FEC 通道 12-15、FEC 通道 16-19、FEC 通道 20-23、FEC 通道 24-27 或 FEC 通道 28-31 可分别认为是第 0 路、第 1 路、第 2 路、第 3 路、第 4 路、第 5 路、第 6 路或第 7 路 100G 中的 FEC 通道 0-3。发端处理模块接着对 32 个通道数据流进行通道纠偏处理（lane de-skew）得到对齐的 32 条通道

数据流。然后根据 FEC 通道 0-3、FEC 通道 4-7、FEC 通道 8-11、FEC 通道 12-15、FEC 通道 16-19、FEC 通道 20-23、FEC 通道 24-27 或 FEC 通道 28-31 的对齐标识对 4 条通道的数据进行通道重排序（lane reorder）处理，使得 4 条通道的数据能够按照指定的顺序排列。最后使得 32 条通道的数据能够按照指定的顺序排列。一种顺序是跟图 11 一样通道数据流从上往下按从 0 到 31 排序。

经通道重排序的 32 条通道数据流送入所设计的包含卷积交织和复用的处理器进行交织打乱数据顺序处理后送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。

在本实施例中，卷积交织处理采用如图 11 所示的结构，分别对 n=32 条 PCS 通道数据流进行卷积交织得到 n=32 条第一数据流。其中卷积交织器 0、卷积交织器 1、卷积交织器 2、…、卷积交织器 31 采用相同的交织结构。图 19（a）给出了一种卷积交织器结构，其包含 p=2 条延迟线（delay line）。这 2 条延迟线分别包括 Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=2 个符号。也就是，延迟线 0 的延迟取值为 2Q 个符号，延迟线 1 的延迟取值为 0 个符号即无延迟。

如图 19(a)所示的$C_r(\cdot)$表示该通道数据流$r(0 \le r \le n-1)$中一个 RS 符号。比如$C_r(4t)$，$C_r(4t+1)$表示通道数据流中当前输入给延迟线 0 的两个 RS 符号且$C_r(4t-4Q)$，$C_r(4t-4Q+1)$为延迟线 0 输出的两个 RS 符号；$C_r(4t+2)$，$C_r(4t+3)$表示通道数据流中接下来输入给延迟线 1 的两个 RS 符号且$C_r(4t+2)$，$C_r(4t+3)$为延迟线 1 输出的两个 RS 符号；$C_r(4t+4)$，$C_r(4t+5)$表示通道数据流中再接下来输入给延迟线 0 的两个 RS 符号且$C_r(4t-4Q+4)$，$C_r(4t-4Q+5)$为延迟线 0 输出的两个 RS 符号，依次类推。结合图 8，可以看出当$4Q+2 \ge 272$即$Q \ge 68$时，卷积交织输出的连续 4 个 RS 符号$C_r(4t-4Q)$，$C_r(4t-4Q+1)$，$C_r(4t+2)$，$C_r(4t+3)$来自 4 个不同的 RS 码字。

图 23 为本申请实施例中卷积交织器的第十一种结构示意图。如图 23 所示，在一种可能的实施方式中，选取$Q=68$，卷积交织器具体结构如图 23 所示。其对应的交织时延约为 68*2*2/2=136 个 RS 符号。

采用如图 23 所示的卷积交织器分别对 32 条 PCS 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 8 所示的 PCS 通道数据流，不难理解，通道数据流 0-3 中的任一个 RS 符号，通道数据流 4-7 中的任一个 RS 符号，通道数据流 8-11 中的任一个 RS 符号，通道数据流 12-15 中的任一个 RS 符号，通道数据流 16-19 中的任一个 RS 符号，通道数据流 20-23 中的任一个 RS 符号，通道数据流 24-27 中的任一个 RS 符号，与通道数据流 28-31 中的任一个 RS 符号，均来自不同的 RS 码字。因此 32 条第一数据流包含 G=8 个第一数据流子集，第一数据流 0-3 为第一数据流子集 0，第一数据流 4-7 为第一数据流子集 1，第一数据流 8-11 为第一数据流子集 2，第一数据流 12-15 为第一数据流子集 3，第一数据流 16-19 为第一数据流子集 4，第一数据流 20-23 为第一数据流子集 5，第一数据流 24-27 为第一数据流子集 6，第一数据流 28-31 为第一数据流子集 7。

本实施例采用如图 13 所示复用处理的一种可能实现方式为 G=8, K=4, m=8，其包含 8 个 4:1 复用器，每个复用器将 4 条第一数据流进行复用得到 1 条第二数据流，总共生成 8 条第二数

据流。从第一数据流子集 0-7 中任意 4 个第一数据流子集各自任取 1 条第一数据流作为 4:1 复用器的输入，一种具体实现方式如图 17（b）所示，4:1 复用器i（$0 \le i \le 7$）的复用输入数据流 0 至 3 分别对应为第 一数据流i，第 一数据流$i+16$，第 一数据流$i+8$，第 一数据流$i+24$。图中C_j^x表示 4:1 复用器的复用输入数据流 j（$0 \le j \le 3$）中连续的$\Delta=4$个 RS 符号，其来自4个不同的外码 RS 码字。4:1 复用器将 4 条输入数据流中数据按照4个 RS 符号为单位以轮询方式输出到输出数据流中，即输出数据顺序为$C_0^0, C_1^0, C_2^0, C_3^0, C_0^1, C_1^1, C_2^1, C_3^1, \ldots, C_0^w, C_1^w, C_2^w, C_3^w \cdots$。输出数据流中连续 16 个 RS 符号来自 16 个不同的 RS 码字。需要说明的是由于$K \le G$，则$\Delta=1,2$仍可以使得第二数据流中连续 16 个 RS 符号来自 16 个不同的 RS 码。

图 24 为本申请实施例中复用器的第六种结构示意图。如图 24 所示，复用器i（$0 \le i \le 3$）将第一数据流i，第一数据流$i+8$，第一数据流$i+16$，第一数据流$i+24$，分别对应于 4:1 复用器的复用输入数据流 0 至 3。图中C_j^x表示 4:1 复用器的复用输入数据流 j（$0 \le j \le 3$）中连续的$\Delta=4$个 RS 符号，其来自4个不同的外码 RS 码字。4:1 复用器将4条输入数据流中数据按照4个 RS 符号为单位以轮询方式输出到输出数据流中，即输出数据顺序为$C_0^0, C_1^0, C_2^0, C_3^0, C_0^1, C_1^1, C_2^1, C_3^1, \ldots, C_0^w, C_1^w, C_2^w, C_3^w \cdots$。输出数据流中连续 16 个 RS 符号来自 16 个不同的 RS 码字。需要说明的是由于$K \le G$，则$\Delta=1,2$仍可以使得第二数据流中连续 16 个 RS 符号来自 16 个不同的 RS 码。

上述 8 条第二数据流分别进行内码编码，其内码编码方案可以采用实施例 1 给出的编码方案，得到与实施例 1 相当的性能；也可以采用实施例 3 给出的编码方案，得到与实施例 3 相当的性能，此处不再赘述。

实施例 13：应用于 8×100G 接口的场景，内码编码信息位长度为 120 比特，采用 2:1 的复用器，采用通道纠偏处理。

在实施例 12 的基础上，本实施例考虑复用处理采用 2:1 复用器，且信息长度为 120 比特的内码，对应地采用新设计的卷积交织器和复用处理。

在本实施例中，卷积交织处理采用如图 11 所示的结构，分别对 n=32 条 PCS 通道数据流进行卷积交织得到 n=32 条第一数据流。其中卷积交织器 0、卷积交织器 1、卷积交织器 2、…、卷积交织器 31 采用相同的交织结构。图 16（a）给出了一种卷积交织器结构，其包含 p=3 条延迟线（delay line）。这 3 条延迟线分别包括 2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=2 个符号。也就是，延迟线 0 的延迟取值为 4Q 个符号，延迟线 1 的延迟取值为 2Q 个符号，延迟线 2 的延迟取值为 0 个符号即无延迟。

如图16(a)所示的$C_r(\cdot)$表示该通道数据流r（$0 \le r \le n-1$）中一个RS符号。比如，$C_r(6t)$，$C_r(6t+1)$表示通道数据流中当前输入给延迟线0的两个RS符号且$C_r(6t-12Q)$，$C_r(6t-12Q+1)$为延迟线 0 输出的两个 RS 符号；$C_r(6t+2)$，$C_r(6t+3)$表示通道数据流中接下来输入给延迟线 1 的两个 RS 符号且$C_r(6t-6Q+2)$，$C_r(6t-6Q+3)$为延迟线 1 输出的两个 RS 符号；$C_r(6t+4)$，$C_r(6t+5)$表示通道数据流中再接下来输入给延迟线 2 的两个 RS 符号且$C_r(6t+4)$，$C_r(6t+5)$为延迟线 2 输出的两个 RS 符号；$C_r(6t+6)$，$C_r(6t+7)$表示通道数据流中再随后输入给延迟线 0 的两个 RS 符号且$C_r(6t-12Q+6)$，$C_r(6t-12Q+7)$为延迟线 0 输出的两个 RS 符号，依次类推。

结合图 8，可以看出当$6Q+2\geq 272$即$Q\geq 45$时，卷积交织输出的$C_r(6t-12Q)$，$C_r(6t-12Q+1)$，$C_r(6t-6Q+2)$，$C_r(6t-6Q+3)$，$C_r(6t+4)$，$C_r(6t+5)$总共 6 个 RS 符号来自 6 个不同的 RS 码字。

图 25 为本申请实施例中卷积交织器的第十二种结构示意图。如图 25 所示，在一种可能的实施方式中，选取$Q=45$，卷积交织器具体结构如图 25 所示。其对应的交织时延约为 90*2*3/2=270 个 RS 符号。

采用如图 25 所示的卷积交织器分别对 32 条 PCS 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 8 所示的 PCS 通道数据流，不难理解，第一数据流 0-3 中的任一个 RS 符号，第一数据流 4-7 中的任一个 RS 符号，第一数据流 8-11 中的任一个 RS 符号，第一数据流 12-15 中的任一个 RS 符号，第一数据流 16-19 中的任一个 RS 符号，第一数据流 20-23 中的任一个 RS 符号，第一数据流 24-27 中的任一个 RS 符号，与第一数据流 28-31 中的任一个 RS 符号，均来自不同的 RS 码字。因此 32 条第一数据流包含 G=8 个第一数据流子集，第一数据流 0-3 为第一数据流子集 0，第一数据流 4-7 为第一数据流子集 1，第一数据流 8-11 为第一数据流子集 2，第一数据流 12-15 为第一数据流子集 3，第一数据流 16-19 为第一数据流子集 4，第一数据流 20-23 为第一数据流子集 5，第一数据流 24-27 为第一数据流子集 6，第一数据流 28-31 为第一数据流子集 7。

本实施例采用如图 13 所示复用处理的一种可能实现方式为 G=8,K=2,m=16,即包含 16 个 2:1 复用器，每个复用器将 2 条第一数据流进行复用得到 1 条第二数据流，总共生成 16 条第二数据流。从第一数据流子集 0-7 中任意 2 个第一数据流子集各自任取 1 条第一数据流作为 2:1 复用器的输入，对应的 2:1 复用器一种具体实现方式如图 17（a）所示，2:1 复用器i（$0\leq i\leq 15$）的复用输入数据流 0 和复用输入数据流 1 分别对应为第一数据流i，第一数据流$i+16$。图中C_j^x表示 2:1 复用器的复用输入数据流 j（$0\leq j\leq 1$）中连续的$\Delta=6$个 RS 符号，其来自6个不同的外码 RS 码字。2:1 复用器将 2 条输入数据流中数据按照6个 RS 符号为单位以轮询方式输出到输出数据流中，即输出数据顺序为$C_0^0, C_1^0, C_0^1, C_1^1, \cdots, C_0^w, C_1^w, \cdots$。输出数据流中连续 12 个 RS 符号来自 12 个不同的 RS 码字。需要说明的是由于$K\leq G$,则$\Delta=1,2,3$仍可以使得第二数据流中连续 12 个 RS 符号来自 12 个不同的 RS 码。

对上述复用处理输出的 16 条第二数据流进行编码方案可以均采用实施例 1 中的方案，得到与实施例 1 相当的性能，此处不再赘述。

实施例 14：应用于 8×100G 接口的场景，内码编码信息位长度为 160 比特，采用 2:1 的复用器，采用通道纠偏处理。

在实施例 12 的基础上，本实施例考虑复用处理采用 2:1 复用器，且信息长度为 160 比特的内码，对应地采用新设计的卷积交织器和复用处理。

在本实施例中，卷积交织处理采用如图 11 所示的结构，分别对 n=32 条 PCS 通道数据流进行卷积交织得到 n=32 条第一数据流。其中卷积交织 0、卷积交织 1、卷积交织 2、…、卷积交织 31 采用相同的交织结构。图 18（a）给出了一种卷积交织器结构，其包含 p=4 条延迟线（delay line）。这 4 条延迟线分别包括 3Q 个存储单元、2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=2 个符号。也就是，延迟线 0 的延迟取值为 6Q 个符

号，延迟线 1 的延迟取值为 4Q 个符号，延迟线 2 的延迟取值为 2Q 个符号，延迟线 3 的延迟取值为 0 个符号即无延迟。

如图 18（a）所示的$C_r(\cdot)$表示该通道数据流$r(0 \le r \le n-1)$中一个 RS 符号。比如，$C_r(8t)$，$C_r(8t+1)$表示通道数据流中当前输入给延迟线 0 的两个 RS 符号且$C_r(8t-24Q)$，$C_r(8t-24Q+1)$为延迟线 0 输出的两个 RS 符号；$C_r(8t+2)$，$C_r(8t+3)$表示通道数据流中接下来输入给延迟线 1 的两个 RS 符号且$C_r(8t-16Q+2)$，$C_r(8t-16Q+3)$为延迟线 1 输出的两个 RS 符号；$C_r(8t+4)$，$C_r(8t+5)$表示通道数据流中再接下来输入给延迟线 2 的两个 RS 符号且$C_r(8t-8Q+4)$，$C_r(8t-8Q+5)$为延迟线 2 输出的两个 RS 符号；$C_r(8t+6)$，$C_r(8t+7)$表示通道数据流中再随后输入给延迟线 3 的两个 RS 符号且$C_r(8t+6)$，$C_r(8t+7)$为延迟线 3 输出的两个 RS 符号，$C_r(8t+8)$，$C_r(8t+9)$表示通道数据流中再随后输入给延迟线 0 的两个 RS 符号且$C_r(8t-24Q+8)$，$C_r(8t-24Q+9)$为延迟线 0 输出的两个 RS 符号，依次类推。结合图 8，可以看出当$8Q+2 \ge 272$即$Q \ge 34$时，卷积交织输出的$C_r(8t-24Q)$，$C_r(8t-24Q+1)$，$C_r(8t-16Q+2)$，$C_r(8t-16Q+3)$，$C_r(8t-8Q+4)$，$C_r(8t-8Q+5)$，$C_r(8t+6)$，$C_r(8t+7)$总共 8 个 RS 符号来自 8 个不同的 RS 码字。

图 26 为本申请实施例中卷积交织器的第十三种结构示意图。如图 26 所示，在一种可能的实施方式中，选取$Q=34$，卷积交织器具体结构如图 26 所示。其对应的交织时延约为 102*2*4/2=408 个 RS 符号。

采用如图 26 所示的卷积交织器分别对 32 条 PCS 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 8 所示的 PCS 通道数据流，不难理解，第一数据流 0-3 中的任一个 RS 符号，第一数据流 4-7 中的任一个 RS 符号，第一数据流 8-11 中的任一个 RS 符号，第一数据流 12-15 中的任一个 RS 符号，第一数据流 16-19 中的任一个 RS 符号，第一数据流 20-23 中的任一个 RS 符号，第一数据流 24-27 中的任一个 RS 符号，与第一数据流 28-31 中的任一个 RS 符号，均来自不同的 RS 码字。因此 32 条第一数据流包含 G=8 个第一数据流子集，第一数据流 0-3 为第一数据流子集 0，第一数据流 4-7 为第一数据流子集 1，第一数据流 8-11 为第一数据流子集 2，第一数据流 12-15 为第一数据流子集 3，第一数据流 16-19 为第一数据流子集 4，第一数据流 20-23 为第一数据流子集 5，第一数据流 24-27 为第一数据流子集 6，第一数据流 28-31 为第一数据流子集 7。

本实施例采用如图 13 所示复用处理的一种可能实现方式为 G=8，K=2，m=16，即包含 16 个 2:1 复用器，每个复用器将 2 条第一数据流进行复用得到 1 条第二数据流，总共生成 16 条第二数据流。从第一数据流子集 0-7 中任意 2 个第一数据流子集各自任取 1 条第一数据流作为 2:1 复用器的输入，对应的 2:1 复用器一种具体实现方式如图 17（a）所示，2:1 复用器i（$0 \le i \le 15$）的复用输入数据流 0 和复用输入数据流 1 分别对应为第一数据流i，第一数据流$i+16$。图中C_j^x表示 2:1 复用器的复用输入数据流 j（$0 \le j \le 1$）中连续的$\Delta=8$个 RS 符号，其来自8个不同的外码 RS 码字。2:1 复用器将 2 条输入数据流中数据按照8个 RS 符号为单位以轮询方式输出到输出数据流中，即输出数据顺序为$C_0^0, C_1^0, C_0^1, C_1^1, \cdots, C_0^w, C_1^w, \cdots$。输出数据流中连续 16 个 RS 符号来自 16 个不同的 RS 码字。需要说明的是由于$K \le G$，则$\Delta=1,2,4$仍可以使得第二数据流中连续 16 个 RS 符号来自 16 个不同的 RS 码。

对上述复用处理输出的 16 条第二数据流进行内码编码方案可采用实施例 3 中的内码编码

方案，得到与实施例 3 相当的性能，此处不再赘述。

实施例 15：应用于 8×100G 接口的场景，采用通道符号对齐。

在实施例 11-14 中任意实施例的基础上，本实施例给出了一种更低时延的实现方案。

基于图 3（b）所示的发端处理模块的数据处理示意图，发端处理模块利用 FEC 通道 0-3、FEC 通道 4-7、FEC 通道 8-11、FEC 通道 12-15、FEC 通道 16-19、FEC 通道 20-23、FEC 通道 24-27 或 FEC 通道 28-31 已知的对齐标识（Alignment marker）进行 4 个通道数据流的标识锁定（alignment lock）。发端处理模块接着对 32 个通道数据流采用基于 2 个 RS 符号对齐得到对齐的 32 条通道数据流。32 条通道数据流分别进行卷积交织得到 32 条第一数据流，第一数据流进行复用处理得到 4 条或者 8 条、或者 16 条第二数据流，并送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。

本实施例中当采用的包含卷积交织和复用的处理器，以及内码编码跟实施例 11 中的方案一样时，其级联码在 AWGN 下性能与其相当，但在对抗系统突发错误略差。该方案适用于一些要求更低时延的场景。

本实施例中当采用的包含卷积交织和复用的处理器，以及内码编码跟实施例 12 中的方案一样时，其级联码在 AWGN 下性能与其相当，但在对抗系统突发错误略差。该方案适用于一些要求更低时延的场景。

本实施例中当采用的包含卷积交织和复用的处理器，以及内码编码跟实施例 13 中的方案一样时，其级联码在 AWGN 下性能与其相当，但在对抗系统突发错误略差。该方案适用于一些要求更低时延的场景。

本实施例中当采用的包含卷积交织和复用的处理器，以及内码编码跟实施例 14 中的方案一样时，其级联码在 AWGN 下性能与其相当，但在对抗系统突发错误略差。该方案适用于一些要求更低时延的场景。

实施例 16：应用于 8×100G 接口的场景，内码编码信息位长度为 120 比特或 160 比特，采用 8:1 的复用器，采用通道纠偏处理。

本实施例考虑客户侧接口为每个通道 100 Gb/s 的 8×100G 且采用“100G RS-FEC”模式，其接口细节见《IEEE Std 802.3ckTM/D3.0》。

基于图 3（a）所示的发端处理模块的数据处理示意图，发端处理模块利用 FEC 通道 0-3、FEC 通道 4-7、FEC 通道 8-11、FEC 通道 12-15、FEC 通道 16-19、FEC 通道 20-23、FEC 通道 24-27 或 FEC 通道 28-31 已知的对齐标识（Alignment marker）进行 4 个通道数据流的标识锁定（alignment lock）。这里 FEC 通道 0-3、FEC 通道 4-7、FEC 通道 8-11、FEC 通道 12-15、FEC 通道 16-19、FEC 通道 20-23、FEC 通道 24-27 或 FEC 通道 28-31 可分别认为是第 0 路、第 1 路、第 2 路、第 3 路、第 4 路、第 5 路、第 6 路或第 7 路 100G 中的 FEC 通道 0-3。发端处理模块接着对 32 个通道数据流进行通道纠偏处理（lane de-skew）得到对齐的 32 条通道数据流。然后根据 FEC 通道 0-3、FEC 通道 4-7、FEC 通道 8-11、FEC 通道 12-15、FEC 通道 16-19、FEC 通道 20-23、FEC 通道 24-27 或 FEC 通道 28-31 的对齐标识对 4 条通道的数据进

行通道重排序（lane reorder）处理，使得 4 条通道的数据能够按照指定的顺序排列。最后使得 32 条通道的数据能够按照指定的顺序排列。一种顺序是跟图 9 一样通道数据流从上往下按从 0 到 31 排序。

经通道重排序的 32 条通道数据流送入所设计的包含卷积交织和复用的处理器进行交织打乱数据顺序处理后送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。

在本实施例中，卷积交织处理采用如图 11 所示的结构，分别对 n=32 条 FEC 通道数据流进行卷积交织得到 n=32 条第一数据流。其中卷积交织器 0、卷积交织器 1、卷积交织器 2、…、卷积交织器 31 采用相同的交织结构。

图 27（a）为本申请实施例中卷积交织器的第十四种结构示意图。如图 27（a）所示，其包含 p=2 条延迟线（delay line）。这 2 条延迟线分别包括 Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=1 个符号。也就是说，延迟线 0 的延迟取值为 Q 个符号，延迟线 1 的延迟取值为 0 个符号即无延迟。

如图27(a)所示的$C_r(\cdot)$表示该通道数据流$r(0 \le r \le n-1)$中一个RS符号。比如，$C_r(2t)$表示通道数据流中当前输入给延迟线 0 的 1 个 RS 符号且$C_r(2t-2Q)$为延迟线 0 输出的 1 个 RS 符号；$C_r(2t+1)$表示通道数据流中接下来输入给延迟线 1 的 1 个 RS 符号且$C_r(2t+1)$为延迟线 1 输出的 1 个 RS 符号；$C_r(2t+2)$表示通道数据流中再接下来输入给延迟线 0 的 1 个 RS 符号且$C_r(2t-2Q+2)$为延迟线 0 输出的 1 个 RS 符号，依次类推。结合图 9，可以看出当$2Q+1 \ge 136$即$Q \ge 68$时，卷积交织输出的连续 2 个 RS 符号$C_r(2t-2Q)$，$C_r(2t+1)$来自 2 个不同的 RS 码字。

图 27（b）为本申请实施例中卷积交织器的第十五种结构示意图。如图 27（b）所示，在一种可能的实施方式中，选取$Q=68$，卷积交织器具体结构如图 27（b）所示。其对应的交织时延约为 68*2/2=68 个 RS 符号。

采用如图 27（b）所示的卷积交织器分别对 32 条 FEC 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 9 所示的 FEC 通道数据流，不难理解，第一数据流 0-3 中的任一 RS 符号，第一数据流 4-7 中的任一 RS 符号，第一数据流 8-11 中的任一 RS 符号，第一数据流 12-15 中的任一 RS 符号，第一数据流 16-19 中的任一 RS 符号，第一数据流 20-23 中的任一 RS 符号，与第一数据流 24-27 中的任一 RS 符号，与第一数据流 28-31 中的任一 RS 符号，均来自不同的 RS 码字。因此 32 条第一数据流包含 G=8 个第一数据流子集，第一数据流 0-3 为第一数据流子集 0，第一数据流 4-7 为第一数据流子集 1，第一数据流 8-11 为第一数据流子集 2，第一数据流 12-15 为第一数据流子集 3，第一数据流 16-19 为第一数据流子集 4，第一数据流 20-23 为第一数据流子集 5，第一数据流 24-27 为第一数据流子集 6，第一数据流 28-31 为第一数据流子集 7。

本实施例采用实施例 11 中的 8:1 复用处理结构，可以得到 4 条第二数据流，且每个第二数据流中连续的 16 个 RS 符号均来自 16 个不同的 RS 码字。

对上述 4 条第二数据流分别进行内码编码，其内码编码方案可以采用实施例 1 给出的编码方案，得到与实施例 1 相当的性能；也可以采用实施例 3 给出的编码方案，得到与实施例 3 相当的性能，此处不再赘述。

实施例 17：应用于 8×100G 接口的场景，内码编码信息位长度为 120 比特，采用 4:1 的复用器，采用通道纠偏处理。

在实施例 16 方案基础上，本实施例给出了内码信息长度为 120 比特，复用处理采用 4:1 复用器时，对应地采用新设计的卷积交织器和复用处理。

在本实施例中，卷积交织处理采用如图 11 所示的结构，分别对 n=32 条 PCS 通道数据流进行卷积交织得到 n=32 条第一数据流。其中卷积交织器 0、卷积交织器 1、卷积交织器 2、…、卷积交织器 31 采用相同的交织结构。

图 28（a）为本申请实施例中卷积交织器的第十六种结构示意图。如图 28（a）所示，其包含 p=3 条延迟线（delay line）。这 p=3 条延迟线分别包括 2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=1 个符号。也就是，延迟线 0 的延迟取值为 2Q 个符号，延迟线 1 的延迟取值为 Q 个符号，延迟线 1 的延迟取值为 0 个符号即无延迟。

如图 28(a)所示的$C_r(\cdot)$表示该通道数据流$r(0 \le r \le n-1)$中一个 RS 符号。比如，$C_r(3t)$表示通道数据流中当前输入给延迟线 0 的 1 个 RS 符号且$C_r(3t-6Q)$为延迟线 0 输出的 1 个 RS 符号；$C_r(3t+1)$表示通道数据流中接下来输入给延迟线 1 的 1 个 RS 符号且$C_r(3t-3Q+1)$为延迟线 1 输出的 1 个 RS 符号；$C_r(3t+2)$表示通道数据流中再接下来输入给延迟线 2 的 1 个 RS 符号且$C_r(3t+2)$为延迟线 2 输出的 1 个 RS 符号；$C_r(3t+3)$表示通道数据流中再随后输入给延迟线 0 的 1 个 RS 符号且$C_r(3t-6Q+3)$为延迟线 0 输出的 1 个 RS 符号，依次类推。结合图 9，可以看出当$3Q+1 \ge 136$即$Q \ge 45$时，卷积交织输出的$C_r(3t-6Q)$，$C_r(3t-3Q+2)$，$C_r(3t+2)$总共 3 个 RS 符号来自 3 个不同的 RS 码字。

图 28（b）为本申请实施例中卷积交织器的第十七种结构示意图。如图 28（b）所示，在一种可能的实施方式中，选取$Q=45$，卷积交织器具体结构如图 28（b）所示。其对应的交织时延约为 90*3/2=135 个 RS 符号。

采用如图 28（b）所示的卷积交织器分别对 32 条 FEC 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 9 所示的 FEC 通道数据流，不难理解，第一数据流 0-3 中的任一 RS 符号，第一数据流 4-7 中的任一 RS 符号，第一数据流 8-11 中的任一 RS 符号，第一数据流 12-15 中的任一 RS 符号，第一数据流 16-19 中的任一 RS 符号，第一数据流 20-23 中的任一 RS 符号，与第一数据流 24-27 中的任一 RS 符号，与第一数据流 28-31 中的任一 RS 符号，均来自不同的 RS 码字。因此 32 条第一数据流包含 G=8 个第一数据流子集，第一数据流 0-3 为第一数据流子集 0，第一数据流 4-7 为第一数据流子集 1，第一数据流 8-11 为第一数据流子集 2，第一数据流 12-15 为第一数据流子集 3，第一数据流 16-19 为第一数据流子集 4，第一数据流 20-23 为第一数据流子集 5，第一数据流 24-27 为第一数据流子集 6，第一数据流 28-31 为第一数据流子集 7。

本实施例采用实施例 12 中的复用处理结构，且$\Delta=1,3$可以得到 8 条第二数据流，且每个第二数据流中连续的 12 个 RS 符号均来自 12 个不同的 RS 码字。

对上述复用处理输出的 8 条第二数据流进行编码方案可以均采用实施例 1 中的方案，得到与实施例 1 相当的性能，此处不再赘述。

实施例 18：应用于 8×100G 接口的场景，内码编码信息位长度为 160 比特，采用 4:1 的复用器，采用通道纠偏处理。

在实施例 16 方案基础上，本实施例给出了内码信息长度为 160 比特，复用处理采用 4:1 复用器时一种次低时延的实现方案，对应地采用新设计的交织器和复用处理。

在本实施例中，卷积交织处理采用如图 11 所示的结构，分别对 n=32 条 PCS 通道数据流进行卷积交织得到 n=32 条第一数据流。其中卷积交织器 0、卷积交织器 1、卷积交织器 2、…、卷积交织器 31 采用相同的交织结构。

图 29（a）为本申请实施例中卷积交织器的第十八种结构示意图。如图 29（a）所示，其包含 p=4 条延迟线（delay line）。这 p=4 条延迟线分别包括 3Q 个存储单元，2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=1 个符号。也就是，延迟线 0 的延迟取值为 3Q 个符号，延迟线 1 的延迟取值为 2Q 个符号，延迟线 2 的延迟取值为 Q 个符号、延迟线 3 的延迟取值为 0 个符号即无延迟。

如图29(a)所示的$C_r(\cdot)$表示该通道数据流$r(0 \le r \le n-1)$中一个RS符号。比如，$C_r(4t)$表示通道数据流中当前输入给延迟线 0 的 1 个 RS 符号且$C_r(4t-12Q)$为延迟线 0 输出的 1 个 RS 符号；$C_r(4t+1)$表示通道数据流中接下来输入给延迟线 1 的 1 个 RS 符号且$C_r(4t-8Q+1)$为延迟线 1 输出的 1 个 RS 符号；$C_r(4t+2)$表示通道数据流中再接下来输入给延迟线 2 的 1 个 RS 符号且$C_r(4t-4Q+2)$为延迟线 2 输出的 1 个 RS 符号；$C_r(4t+3)$表示通道数据流中再接下来输入给延迟线 3 的 1 个 RS 符号且$C_r(4t+3)$为延迟线 3 输出的 1 个 RS 符号；$C_r(4t+4)$表示通道数据流中再随后输入给延迟线 0 的 1 个 RS 符号且$C_r(4t-12Q+4)$为延迟线 0 输出的 1 个 RS 符号，依次类推。结合图 9，可以看出当$4Q+1 \ge 136$即$Q \ge 34$时，卷积交织连续输出的$C_r(4t-12Q)$，$C_r(4t-8Q+1)$，$C_r(4t-4Q+2)$，$C_r(4t+3)$总共 4 个 RS 符号来自 4 个不同的 RS 码字。

图 29（b）为本申请实施例中卷积交织器的第十九种结构示意图。如图 29（b）所示，在一种可能的实施方式中，选取$Q=34$，卷积交织器具体结构如图 29（b）所示。其对应的交织时延约为 102*4/2=204 个 RS 符号。

采用如图 29（b）所示的卷积交织器分别对 32 条 FEC 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 9 所示的 FEC 通道数据流，不难理解，第一数据流 0-3 中的任一 RS 符号，第一数据流 4-7 中的任一 RS 符号，第一数据流 8-11 中的任一 RS 符号，第一数据流 12-15 中的任一 RS 符号，第一数据流 16-19 中的任一 RS 符号，第一数据流 20-23 中的任一 RS 符号，与第一数据流 24-27 中的任一 RS 符号，与第一数据流 28-31 中的任一 RS 符号，均来自不同的 RS 码字。因此 32 条第一数据流包含 G=8 个第一数据流子集，第一数据流 0-3 为第一数据流子集 0，第一数据流 4-7 为第一数据流子集 1，第一数据流 8-11 为第一数据流子集 2，第一数据流 12-15 为第一数据流子集 3，第一数据流 16-19 为第一数据流子集 4，第一数据流 20-23 为第一数据流子集 5，第一数据流 24-27 为第一数据流子集 6，第一数据流 28-31 为第一数据流子集 7。

本实施例采用实施例 13 中的复用处理结构，且$\Delta=1,2,4$时可以得到 8 条第二数据流，且每个第二数据流中连续的 16 个 RS 符号均来自 16 个不同的 RS 码字。

上述 8 条第二数据流分别进行内码编码，其内码编码方案可以采用实施例 3 的内码编码方案，可以得到与实施例 3 相当的性能，此处不再赘述。

实施例 19：应用于 8×100G 接口的场景，内码编码信息位长度为 120 比特，采用 2:1 的复用器，采用通道纠偏处理。

在实施例 16 方案基础上，本实施例给出了内码信息长度为 120 比特，复用处理采用 4:1 复用器时，对应地采用新设计的卷积交织器和复用处理。

在本实施例中，卷积交织处理采用如图 11 所示的结构，分别对 n=32 条 PCS 通道数据流进行卷积交织得到 n=32 条第一数据流。其中卷积交织器 0、卷积交织器 1、卷积交织器 2、…、卷积交织器 31 采用相同的交织结构。

图 30（a）为本申请实施例中卷积交织器的第二十种结构示意图。如图 30（a）所示，其包含 p=6 条延迟线（delay line）。这 p=6 条延迟线分别包括 5Q 个存储单元、4Q 个存储单元、3Q 个存储单元、2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=1 个符号。也就是，延迟线 0 的延迟取值为 5Q 个符号，延迟线 1 的延迟取值为 4Q 个符号，延迟线 2 的延迟取值为 3Q 个符号，延迟线 3 的延迟取值为 2Q 个符号，延迟线 4 的延迟取值为 Q 个符号，延迟线 5 的延迟取值为 0 个符号即无延迟。

如图30(a)所示的$C_r(\cdot)$表示该通道数据流$r(0 \le r \le n-1)$中一个RS符号。比如，$C_r(6t)$表示通道数据流中当前输入给延迟线 0 的 1 个 RS 符号且$C_r(6t-30Q)$为延迟线 0 输出的 1 个 RS 符号；$C_r(6t+1)$表示通道数据流中接下来输入给延迟线 1 的 1 个 RS 符号且$C_r(6t-24Q+1)$为延迟线 1 输出的 1 个 RS 符号；$C_r(6t+2)$表示通道数据流中再接下来输入给延迟线 2 的 1 个 RS 符号且$C_r(6t-18Q+2)$为延迟线 2 输出的 1 个 RS 符号；$C_r(6t+3)$表示通道数据流中再接下来输入给延迟线 3 的 1 个 RS 符号且$C_r(6t-12Q+3)$为延迟线 3 输出的 1 个 RS 符号；$C_r(6t+4)$表示通道数据流中再接下来输入给延迟线 4 的 1 个 RS 符号且$C_r(6t-6Q+4)$为延迟线 4 输出的 1 个 RS 符号；$C_r(6t+5)$表示通道数据流中再接下来输入给延迟线 5 的 1 个 RS 符号且$C_r(6t+5)$为延迟线 5 输出的 1 个 RS 符号；$C_r(6t+6)$表示通道数据流中再随后输入给延迟线 0 的 1 个 RS 符号且$C_r(6t-30Q+6)$为延迟线 0 输出的 1 个 RS 符号，依次类推。结合图 9，可以看出当$6Q+1 \ge 136$即$Q \ge 23$时，卷积交织连续输出的$C_r(6t-30Q)$，$C_r(6t-24Q+1)$，$C_r(6t-18Q+2)$，$C_r(6t-12Q+3)$，$C_r(6t-6Q+4)$，$C_r(6t+5)$总共 6 个 RS 符号来自 6 个不同的 RS 码字。

图 30（b）为本申请实施例中卷积交织器的第二十一种结构示意图。如图 30（b）所示，在一种可能的实施方式中，选取$Q=23$，卷积交织器具体结构如图 30（b）所示。其对应的交织时延约为 23*5*6/2=345 个 RS 符号。

采用如图 30（b）所示的卷积交织器分别对 32 条 FEC 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 9 所示的 FEC 通道数据流，不难理解，第一数据流 0-3 中的任一 RS 符号，第一数据流 4-7 中的任一 RS 符号，第一数据流 8-11 中的任一 RS 符号，第一数据流 12-15 中的任一 RS 符号，第一数据流 16-19 中的任一 RS 符号，第一数据流 20-23 中的任一 RS 符号，与第一数据流 24-27 中的任一 RS 符号，与第一数据流 28-31 中的任一 RS 符号，均

来自不同的 RS 码字。因此 32 条第二数据流包含 G=8 个第二数据流子集，第二数据流 0-3 为第二数据流子集 0，第二数据流 4-7 为第二数据流子集 1，第二数据流 8-11 为第二数据流子集 2，第二数据流 12-15 为第二数据流子集 3，第二数据流 16-19 为第二数据流子集 4，第二数据流 20-23 为第二数据流子集 5，第二数据流 24-27 为第二数据流子集 6，第二数据流 28-31 为第二数据流子集 7。

本实施例采用实施例 13 中的复用处理结构，且$\Delta= 1,2,3,6$时可以得到 16 条第二数据流，且每个第二数据流中连续的 12 个 RS 符号均来自 12 个不同的 RS 码字。

上述 16 条第二数据流分别进行内码编码，其编码方案可以采用实施例 1 中的内码编码方案，并可以取得与实施例 1 相当的性能，此处不再赘述。

实施例 20：应用于 8×100G 接口的场景，内码编码信息位长度为 160 比特，采用 2:1 的复用器，采用通道纠偏处理。

在实施例 16 方案基础上，本实施例给出了内码信息长度为 160 比特，复用处理采用 2:1 复用器时，对应地采用新设计的卷积交织器和复用处理。

在本实施例中，卷积交织处理采用如图 11 所示的结构，分别对 n=32 条 PCS 通道数据流进行卷积交织得到 n=32 条第一数据流。其中卷积交织器 0、卷积交织器 1、卷积交织器 2、…、卷积交织器 31 采用相同的交织结构。

图 31（a）为本申请实施例中卷积交织器的第二十二种结构示意图。如图 31（a）所示，其包含 p=8 条延迟线（delay line）。这 p=8 条延迟线分别包括 7Q 个存储单元，6Q 个存储单元，5Q 个存储单元，4Q 个存储单元，3Q 个存储单元，2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=1 个符号。也就是，延迟线 0 的延迟取值为 7Q 个符号，延迟线 1 的延迟取值为 6Q 个符号，延迟线 2 的延迟取值为 5Q 个符号、延迟线 3 的延迟取值为 4Q 个符号、延迟线 4 的延迟取值为 3Q 个符号、延迟线 5 的延迟取值为 2Q 个符号、延迟线 6 的延迟取值为 Q 个符号、延迟线 7 的延迟取值为 0 个符号即无延迟。

如图31(a)所示的$C_r(\cdot)$表示该通道数据流$r(0 \le r \le n-1)$中一个RS符号。比如，$C_r(8t)$表示通道数据流中当前输入给延迟线 0 的 1 个 RS 符号且$C_r(8t-56Q)$为延迟线 0 输出的 1 个 RS 符号；$C_r(8t+1)$表示通道数据流中接下来输入给延迟线 1 的 1 个 RS 符号且$C_r(8t-48Q+1)$为延迟线 1 输出的 1 个 RS 符号；$C_r(8t+2)$表示通道数据流中再接下来输入给延迟线 2 的 1 个 RS 符号且$C_r(8t-40Q+2)$为延迟线 2 输出的 1 个 RS 符号；$C_r(8t+3)$表示通道数据流中再接下来输入给延迟线 3 的 1 个 RS 符号且$C_r(8t-32Q+3)$为延迟线 3 输出的 1 个 RS 符号；$C_r(8t+4)$表示通道数据流中接下来输入给延迟线 4 的 1 个 RS 符号且$C_r(8t-24Q+4)$为延迟线 4 输出的 1 个 RS 符号；$C_r(8t+5)$表示通道数据流中接下来输入给延迟线 5 的 1 个 RS 符号且$C_r(8t-16Q+5)$为延迟线 5 输出的 1 个 RS 符号；$C_r(8t+6)$表示通道数据流中接下来输入给延迟线 6 的 1 个 RS 符号且$C_r(8t-8Q+6)$为延迟线 6 输出的 1 个 RS 符号；$C_r(8t+7)$表示通道数据流中接下来输入给延迟线 7 的 1 个 RS 符号且$C_r(8t+7)$为延迟线 7 输出的 1 个 RS 符号；$C_r(8t+8)$表示通道数据流中再随后输入给延迟线 0 的 1 个 RS 符号且$C_r(8t-56Q+8)$为延迟线 0 输出的 1 个 RS 符号，依次类推。结合图 9，可以看出当$8Q+1 \ge 136$即$Q \ge 17$时，卷积交织连续输出的$C_r(8t-56Q)$，

$C_r(8t-48Q+1), C_r(8t-40Q+2)$，$C_r(8t-32Q+3), C_r(8t-24Q+4)$，
$C_r(8t-16Q+5), C_r(8t-8Q+6), C_r(8t+7)$总共 8 个 RS 符号来自 8 个不同的 RS 码字。

图 31（b）为本申请实施例中卷积交织器的第二十三种结构示意图。如图 31（b）所示，在一种可能的实施方式中，选取$Q = 17$，卷积交织器具体结构如图 31（b）所示。其对应的交织时延约为 17*7*8/2=476 个 RS 符号。

采用如图 31（b）所示的卷积交织器分别对 32 条 FEC 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 9 所示的 FEC 通道数据流，不难理解，第一数据流 0-3 中的任一 RS 符号，第一数据流 4-7 中的任一 RS 符号，第一数据流 8-11 中的任一 RS 符号，第一数据流 12-15 中的任一 RS 符号，第一数据流 16-19 中的任一 RS 符号，第一数据流 20-23 中的任一 RS 符号，与第一数据流 24-27 中的任一 RS 符号，与第一数据流 28-31 中的任一 RS 符号，均来自不同的 RS 码字。因此 32 条第一数据流包含 G=8 个第一数据流子集，第一数据流 0-3 为第一数据流子集 0，第一数据流 4-7 为第一数据流子集 1，第一数据流 8-11 为第一数据流子集 2，第一数据流 12-15 为第一数据流子集 3，第一数据流 16-19 为第一数据流子集 4，第一数据流 20-23 为第一数据流子集 5，第一数据流 24-27 为第一数据流子集 6，第一数据流 28-31 为第一数据流子集 7。

本实施例采用实施例 13 中的复用处理结构，且$\Delta = 1,2,4,8$时可以得到 16 条第二数据流，且每个第二数据流中连续的 16 个 RS 符号均来自 16 个不同的 RS 码字。

上述 16 条第二数据流分别进行内码编码，其内码编码方案可以采用实施例 3 的内码编码方案，可以取得与实施例 3 相当的性能，此处不再赘述。

实施例 21：应用于 8×100G 接口的场景，内码编码信息位长度为 120 比特或 160 比特，采用通道符号对齐。

在实施例 16-20 中任意实施例方案基础上，本实施例给出了一种更低时延的实现方案。

基于图 3（d）所示的发端处理模块的数据处理示意图，发端处理模块利用 FEC 通道 0-3、FEC 通道 4-7、FEC 通道 8-11、FEC 通道 12-15、FEC 通道 16-19、FEC 通道 20-23、FEC 通道 24-27 或 FEC 通道 28-31 已知的对齐标识（Alignment marker）进行 4 个通道数据流的标识锁定（alignment lock）。发端处理模块接着对 32 个通道数据流采用基于 1 个 RS 符号对齐得到对齐的 32 条通道数据流。32 条通道数据流分别进行卷积交织得到 32 条第一数据流，第一数据流进行复用处理得到 4 条或者 8 条、或者 16 条第二数据流，并送入内码编码器进行内码编码。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。

应理解，本实施例采用的复用以及内码编码方案均采用实施例 16-20 中任意实施例给出的方案。

当本实施例中采用的包含复用的处理器以及内码编码跟实施例 16 中的方案一样时，其级联码在 AWGN 下性能与其相当，但在对抗系统突发错误略差。该方案适用于一些要求更低时延的场景。

当本实施例中采用的包含复用的处理器以及内码编码跟实施例 17 中的方案一样时，其级联码在 AWGN 下性能与其相当，但在对抗系统突发错误略差。该方案适用于一些要求更低时延

的场景。

当本实施例中采用的包含复用的处理器以及内码编码跟实施例 18 中的方案一样时，其级联码在 AWGN 下性能与其相当，但在对抗系统突发错误略差。该方案适用于一些要求更低时延的场景。

当本实施例中采用的包含复用的处理器以及内码编码跟实施例 19 中的方案一样时，其级联码在 AWGN 下性能与其相当，但在对抗系统突发错误略差。该方案适用于一些要求更低时延的场景。

当本实施例中采用的包含复用的处理器以及内码编码跟实施例 20 中的方案一样时，其级联码在 AWGN 下性能与其相当，但在对抗系统突发错误略差。该方案适用于一些要求更低时延的场景。

需要说明的是，在一些可能的实施方式中，上述各实施例中介绍的复用处理也可以替换为分组交织来实现，下面结合一个具体实施例进行介绍。

图 32（a）为本申请实施例中对 n 条第一数据流进行分组交织的一种结构示意图。如图 32（a）所示，可以通过 m 个并行的分组交织模块来进行分组交织。具体地，每个分组交织器将输入的 K 条第一数据流通过分组交织后生成一条第二数据流，总共得到 m 条第二数据流。其中，输入到每个分组交织器的 K 条第一数据流的选取方式和上述各实施例中输入复用器的 K 个输入复用数据流的选取方式一致，此处不再赘述。

图 32（b）为本申请实施例中分组交织器的一种结构示意图。如图 32（b）所示，第一数据流i_k中连续Δ个 RS 符号为一个符号子集，用$S_k(.)$表示，其中$0 \le k \le K-1$。因此，$S_k(0), S_k(1), \ldots, S_k(W)$表示第一数据流$i_k$输出的连续$W$个符号子集。第二数据流中连续$\Delta$个 RS 符号为一个符号子集，用 S(.)表示。从输入的 K 条第一数据流中各获取W个符号子集，构成一个 K 行W列的第一符号矩阵，第一符号矩阵中的每个元素为一个符号子集，则$S_k(w)$对应为第一符号矩阵中第k行第w列的元素，其中$0 \le k \le K-1$，$0 \le w \le W-1$。K 行W列的第一符号矩阵通过分组交织后得到一个 1 行 C 列的第二符号矩阵，其中，C=K*W。第二符号矩阵中的每个元素也为一个符号子集，用$S(c)$表示第二符号矩阵中第 c 列的元素，其中$0 \le c \le C$。分组交织$S(c)$与$S_k(w)$之间的映射可以表示为：$c = K * w + k$。分组交织后的第二符号矩阵S(0)，S(0)，…，S(K*W-1) 依次对应为 Q 个内码的信息位，内码编码后得到 Q 个内码，若内码长度为 D 个符号，则有 K*W*Δ=Q*D。

需要说明的是，上述符号子集只是为了便于描述而引入的概念，在实际应用中，第一数据流和第二数据流都分别为一个整体并不存在划分。每个符号子集可以视作第一数据流或第二数据流中的一个或多个符号。另外，在实际应用中，上述的第一符号矩阵和第二符号矩阵也可能不是以矩阵形式呈现的。例如，第一符号矩阵呈现为第一符号集合，第一符号集合包括 K*W 个符号子集，分别对应第一符号矩阵中的 K 行 W 列个元素。第二符号矩阵呈现为第二符号集合，第二符号集合包括 C 个符号子集，分别对应第二符号矩阵中的 1 行 C 列个元素。

作为一个示例，基于实施列 1 的给出的卷积交织器，对应分组交织的一种具体实现为 G=2, K=2, m=16。分组交织器 i 的输入第一数据流i_0和第一数据流i_1分别对应于第一数据流 i 和

第一数据流 i+16。以图 32(b)所示的分组交织器结构为例，分组交织器的参数可以为Δ=6,W=1, Q=1；也可以为Δ=3,W=2, Q=1。第二符号矩阵映射成一个信息位长度是 120 比特的内码的信息数据，可以采用实施列 1 中的内码编码方案，实现相当的性能。

作为另一个示例，基于实施列 1 的给出的卷积交织器，对应分组交织的另一种具体实现为 G=2,K=4,m=16。分组交织器 i 的输入第一数据流i_0至第一数据流i_3分别对应于第一数据流 i、第一数据流 i+16、第一数据流 i+8、第一数据流 i+24。以图 32（b）所示的分组交织器结构为例，分组交织器的参数可以为Δ=6,W=1, Q=2。第二符号矩阵映射成 2 个信息长度位是 120 比特的内码的信息数据，可以采用实施列 1 中的内码编码方案，实现相当的性能。

作为又一个示例，基于实施列 1 的给出的卷积交织器，对应分组交织的又一种具体实现为 G=2,K=8,m=16。分组交织器 i 的输入第一数据流i_0至第一数据流i_7分别对应于第一数据流 i，第一数据流 i+16，第一数据流 i+8，第一数据流 i+24，第一数据流 i+4，第一数据流 i+20，第一数据流 i+12，第一数据流 i+28。以图 32（b）所示的分组交织器结构为例，分组交织器的参数可以为Δ=6,W=1, Q=4。第二符号矩阵映射成 4 个信息位长度是 120 比特的内码的信息数据，可以采用实施列 1 中的内码编码方案，实现相当的性能。

应理解，基于其他实施列的卷积交织和内码编码方案也可以推导出对应的分组交织的实现方式，此处不再一一列举。

上面对本申请实施例提供的数据处理方法进行了介绍，下面介绍本申请实施例提供的数据处理装置。

图 33 为本申请实施例中数据处理装置的一种结构示意图。如图 33 所示，该数据处理装置包括卷积交织器 101 和复用器 102。卷积交织器 101 用于执行上述数据处理方法中对 n 条通道数据流分别进行卷积交织得到 n 条第一数据流的操作。复用器 201 用于执行上述数据处理方法中对 n 条第一数据流中的每 K 条第一数据流进行复用得到 1 条第二数据流，以得到共 m 条所述第二数据流的操作。具体可以参照上述数据处理方法中关于卷积交织操作和复用操作的相关介绍，此处不再赘述。

应理解，本申请提供的装置也可以通过其他方式实现。例如，上述装置中的单元划分仅仅是一种逻辑功能划分，实际实现时可以有另外的划分方式，例如，多个单元或组件可以结合或可以集成到另一个系统。另外，本申请各个实施例中的各功能单元可以集成在一个处理单元中，也可以是各个独立的物理单元，也可以是两个或两个以上个功能单元集成在一个处理单元中。上述集成的单元既可以采用硬件的形式实现，也可以采用软件功能单元的形式实现。

需要说明的是，除了上述实施例介绍的数据处理方法外，本申请还提供了另一种数据处理方法，下面进行详细介绍。

本实施例中，先对 n 条通道数据流进行交织处理得到 m 条目标数据流。再对 m 条目标数据流分别进行第二 FEC 编码得到编码数据流。其中，对 m 条目标数据流分别进行第二 FEC 编码的方式与上述图 10 所示实施例中步骤 1003 介绍的方式类似，此处不再赘述。

本实施例中，通道数据流可以是PCS通道数据流或FEC通道数据流，具体此处不做限定。n条通道数据流都是经过第一FEC编码的数据流，也就是上文中介绍的经过外码编码的数据流，其中，n为大于1的整数，且为4的倍数。例如，该外码编码可以采用RS码，经过外码编码后的n条数据流可以包括多个RS码字。在实际应用中也可以采用其他的编码方式进行外码编码，为了便于描述，下文统一用RS码字来表示外码编码后生成的码字。应理解，经过外码编码后的每a个码字分布在b条通道数据流中，其中，a≤b≤n，n能被b整除，a为大于或等于1的整数。在上述图5-图9所示的不同应用场景中，a和b的取值也可能不同。以图5所示的应用场景为例，n=32，a=2，b=16，即每2个码字分布在16条通道数据流中。在图6-图9其他几个应用场景中a和b的取值可以通过附图推导得出，此处不再一一赘述。因此b最大取值为16，最小取值为4。需要说明的是，本申请中外码码长取值是以符号为单位来统计的，其中，符号可以包括一个或多个比特。例如，外码为采用的KP4 RS(544,514)码，码长N=544个符号，一个符号包含10个比特。

应理解，每条目标数据流中连续的F个符号来自F个不同的码字，F＞a。每条目标数据流中连续的F个符号至少来自K1条不同的通道数据流，每条目标数据流中连续的F个符号来自n条通道数据流对齐的n个符号中最多K2个符号，K1和K2为n的约数，K2为K1的约数。其中，n条通道数据流对齐的n个符号指的是位于每条通道数据流同一位置的n个符号，以上述图5-图9所示的不同应用场景为例，n条通道数据流每一列的n个符号是对齐的，即n条通道数据流对齐的n个符号可以是n条通道数据流其中一例的n个符号。还应理解，每条目标数据流中连续的F个符号中最多有K3个符号来自相同的通道数据流，K3=⌈F/K1⌉，⌈F/K1⌉表示对F/K1的商进行向上取整的整数，K3个符号中任意两个符号在同一通道数据流上相隔至少K4个符号，K4≥a*N*K2/n，其中N为码字长度。在本实施列中b最大取值为16，最小取值为4，因此在一些可能的实施方式中，K1=n/4，K2=n/16。

在一些可能的实施方式中，在对n条通道数据流进行交织处理得到m条目标数据流之前，还可以对n路通道数据流进行通道重排序，以使得n路数据流按照预设顺序排列。以32条数据流为例，32条数据流可以是按照0到31排序从上到下排列，当然可以简单扩展到其他顺序排列，其具体实现方式本领域普通技术人员可知，此处不再赘述。

在一些可能的实施方式中，在对n条通道数据流进行交织处理得到m条目标数据流之前，还可以对n条通道数据流进行通道数据对齐处理。该通道数据对齐处理可以是采用现有标准定义的通道纠偏处理（lane de-skew），使得其输出的n条通道数据流的数据是完全对齐的。或者，上述“通道数据对齐处理”也可以仅是通道符号对齐，使得其输出的n条通道数据流上的数据是基于外码符号对齐，具体可以是基于一个外码符号对齐，也可以是基于多个外码符号对齐。关于通道数据对齐处理的详细介绍可以参考上述图3（e）的相关描述，此处不再赘述。

需要说明的是，本实施例中的交织处理可以有多种具体的实现方式，下面分别进行介绍。

图34为本申请实施例提供的交织处理的一种流程示意图。

3401、对n条通道数据流分别进行卷积交织得到n条第一数据流。

本实施例中，经过卷积交织后每条第一数据流中连续的 z 个符号来自至少 e 个不同的码字，其中，z 为大于 1 的整数，a≤e≤F，e*k2≥F。每条第一数据流中连续的 z 个符号中最多 k1/k2 个符号来自同一个所述码字。

需要说明的是，本实施例中卷积交织的实现方式与上述图 10 所示实施例中介绍的卷积交织方式类似，此处不再赘述。在一种可能的实施方式中，若采用上述图 12（a）所示的卷积交织器，则卷积交织器的参数应满足$d(p*Q+1)\geq K4$，以使得卷积交织器输出的第一数据流中连续的 z=p*d 个符号至少来自 e 个不同的外码码字，其中，N 为外码编码的码字长度。在另一种可能的实施方式中，若采用上述图 12（b）所示的卷积交织器，则卷积交织器的参数应满足$d(p*Q-1)\geq K4$，以使得卷积交织器输出的第一数据流中连续的 z=p*d 个符号至少来自 e 个不同的外码码字，其中，N 为外码编码的码字长度。

3402、对 n 条第一数据流中的每 K1 条第一数据流进行分组交织得到 S 条目标数据流，以得到共 m 条目标数据流。

图 35 为本申请实施例中对 n 条第一数据流进行分组交织的一种结构示意图。如图 35 所示，可以采用 T 个分组交织器进行分组交织处理。具体地，n 条第一数据流中的每 K1 条第一数据流输入一个分组交织器，该分组交织器输出 S 条目标数据流。T 个分组交织器将输出共 m 条目标数据流，其中，m=S*T,T=n/K1，S≥k1/k2。需要说明的是，n 条第一数据流包括 K1 个第一数据流分组，每个第一数据流分组中各取一条第一数据流作为参与分组交织的 K1 条第一数据流，其中，同一第一数据流分组中任意两条第一数据流的符号来自相同的码字。

应理解，上述第一数据流分组只是为了便于描述而引入的概念，在实际应用中，n 条第一数据流为一个整体并不存在划分，每个第一数据流分组可以视作 n 条第一数据流中的一条或多条数据流。

下面介绍分组交织的一种具体实现方式。

图 36 为本申请实施例中进行分组交织的一种实现方式示意图。如图 36 所示，以其中一个分组交织器的实现方式为例进行介绍。参与分组交织的 K 条第一数据流中每条第一数据流包括 B 个符号，即该 K 条第一数据流包括第一符号矩阵，该第一符号矩阵包括 K 行 B 列个符号。其中，B=R*p*d，R 为大于或等于 1 的整数，第一符号矩阵每行的 B 个符号是由卷积交织器的 p 条延迟线轮询 R 次输出的连续 B 个符号。经过分组交织后的 S 条目标数据流中每条目标数据流包括 F 个符号，即该 S 条目标数据流包括第二符号矩阵，第二符号矩阵包括 S 行 F 列个符号。其中，K*B=S*F，F 为内码信息数据的长度。应理解，第一符号矩阵中的符号至少来自 F 个不同码字，第一符号矩阵中最多有 R*K1/K2 个符号来自相同的码字，第一符号矩阵中选 F 个来自不同码字的符号映射到第二符号矩阵的一行。即每条目标数据流中连续的 F 个符号来自 F 个不同的码字。

在一种可能的实施方式中，第二符号矩阵中每行的 F 个符号来自第一符号矩阵中至少$\lfloor F/K2\rfloor$列，且$\lfloor F/K2\rfloor$列中每列最多取$K2$个符号，其中，$\lfloor F/K2\rfloor$表示对$F/K2$的商进行向上取整的整数。第二符号矩阵中每行的 F 个符号包括第一符号矩阵中每行的至少$\lfloor F/K1\rfloor$个符号，$\lfloor F/K1\rfloor$表示对 F/K1 的商进行向下取整的整数，且第二符号矩阵中每行的 F 个符号包括第一符号矩阵中每行的最多$\lfloor F/K1\rfloor$个符号，$\lfloor F/K1\rfloor$表示对 F/K1 的商进行向上取整的整数。

在另一种可能的实施方式中，第二符号矩阵的每一行中来自第一符号矩阵奇数列的符号分别位于第一符号矩阵的不同行，第二符号矩阵的每一行中来自第一符号矩阵偶数列的符号位于第一符号矩阵的不同行。

在又一种可能的实施方式中，第二符号矩阵的每一行中由延迟取值相同的延迟线输出的符号来自第一符号矩阵的不同行。第二符号矩阵的每一行的最多 K3 个符号来自第一符号矩阵的同一行，该K3个符号中任意2个符号分别由延迟差值大于或等于2*Q*d的2条延迟线输出。

需要说明的是，在实际应用中，上述的第一符号矩阵和第二符号矩阵也可能不是以矩阵形式呈现的。例如，第一符号矩阵呈现为第一符号集合，第一符号集合包括 K*B 个符号子集，分别对应第一符号矩阵中的 K 行 B 列个符号。第二符号矩阵呈现为第二符号集合，第二符号集合包括 S*F 个符号子集，分别对应第二符号矩阵中的 S 行 F 列个符号。

本发明针对采用级联 FEC 的传输方案，设计一种包含卷积交织、分组交织和编码的数据交织编码方法，可以对所有接入业务采用同样的交织方案，且使得整体级联 FEC 方案性能较好且时延较低。使得该级联 FEC 传输方案能够应用于较多的传输场景，尤其适用于对需要较低传输时延的传输场景，如低时延数据中心互联场景。

下面结合一些具体的实施例对上述图 34 所描述的交织处理方法的流程进行进一步介绍。应理解，下述每个实施例中的内码并行度指的是目标数据流的数量，例如，内码并行度 16 表示目标数据流的数量是 16。还应理解，下述各实施例中的目标符号子集均表示目标数据流中连续的 F 个符号。

实施例 1：对于所有业务采用统一交织，内码编码信息位长度为 120 比特，内码并行度 16。

本实施例给出了在 OIF LR 场景下对业务不敏感的具体的数据交织和编码的方案。基发端处理模块根据接入业务的类型，利用各业务 PCS 通道或者 FEC 通道已知的对齐标识（Alignment marker）进行通道数据流的标识锁定（alignment lock）。这里 32 个通道的已知对齐标识各不相同，且与接入的业务相关。发端处理模块接着对每个业务内的多个 PCS 通道或者 FEC 通道进行纠偏处理，纠偏处理后同一个业务内的多个 PCS/FEC 通道之间都是 AM 对齐，而业务之间的通道数据流仅需要满足 RS 符号对齐，即 AM 之间相差整数个 RS 符号。

图 37 为客户侧接口为 2×400GbE 时通道对齐后数据流格式的示意图。比如当接入业务为 2*400GbE 时，纠偏处理后 32 条通道数据流如图 37 所示，每个 400GbE 业务内 16 条 PCS 通道之间都为 AM 对齐，而 2 个业务之间的 AM 相差整数个 RS 符号。然后根据对齐标识对 n=32 条通道的数据进行通道重排序（lane reorder）处理，使得 n=32 条通道的数据能够按照指定的顺序排列。如当接入业务为1*800GbE，32条通道数据流如图5所示，如当接入业务为2*400GbE，32 条通道数据流如图 6 所示，如当接入业务为 4*200GbE，32 条通道数据流如图 7 所示；如当接入业务为 8*100GbE 且为“100G RS-FEC-Int”模式，32 条通道数据流如图 8 所示；如当接入业务为 8*100GbE 且为“100G RS-FEC”模式时，32 条通道数据流如图 9 所示。

经通道重排序的 n=32 条通道数据流送入卷积交织模块和分组交织模块进行相应处理后送入内码编码器进行内码编码；经过内码编码的数据流进行数据处理后送入信道传输媒介进

行传输。该数据处理可包括调制映射（mapping），信道交织（channel interleaving），偏振划分（Polarization distribution），或 DSP 成帧处理（Framing）等。

在本实施例中，分别对 n=32 条 PCS 通道数据流进行卷积交织得到 n=32 条第一数据流。其中卷积交织 0、卷积交织 1、卷积交织 2、…、卷积交织 31 采用相同的交织结构。图 16（a）给出了一种卷积交织器结构，其包含 p=3 条延迟线（delay line）。这 p=3 条延迟线分别包括 2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=2 个符号。也就是，延迟线 0 的延迟取值为 4Q 个符号，延迟线 1 的延迟取值为 2Q 个符号，延迟线 2 的延迟取值为 0 个符号即无延迟。

如图 16(a)所示的 $C_r()$ 表示该通道数据流$r(0 \le r \le n-1)$中一个 RS 符号。比如，$C_r(6t)$，$C_r(6t+1)$表示通道数据流中当前输入给延迟线 0 的两个 RS 符号且$C_r(6t-12Q)$，$C_r(6t-12Q+1)$为延迟线 0 输出的两个 RS 符号；$C_r(6t+2)$，$C_r(6t+3)$表示通道数据流中接下来输入给延迟线 1 的两个 RS 符号且$C_r(6t-6Q+2)$，$C_r(6t-6Q+3)$为延迟线 1 输出的两个 RS 符号；$C_r(6t+4)$，$C_r(6t+5)$表示通道数据流中再接下来输入给延迟线 2 的两个 RS 符号且$C_r(6t+4)$，$C_r(6t+5)$为延迟线 2 输出的两个 RS 符号；$C_r(6t+6)$，$C_r(6t+7)$表示通道数据流中再随后输入给延迟线 0 的两个 RS 符号且$C_r(6t-12Q+6)$，$C_r(6t-12Q+7)$为延迟线 0 输出的两个 RS 符号，依次类推。当接入业务为 1*800GbE 或者 2*400GbE 时，满足 $d(pQ+1) \ge a * \frac{N}{b}$，即$6Q+2 \ge 2*544/16, Q \ge 11$时，卷积交织输出的$C_r(6t-12Q)$，$C_r(6t-12Q+1)$，$C_r(6t-6Q+2)$，$C_r(6t-6Q+3)$，$C_r(6t+4)$，$C_r(6t+5)$总共 6 个 RS 符号来自 6 个不同的 RS 码字。再利用分组交织将 2 个第一数据流子集的数据汇聚可以使得目标数据流连续 12 个 RS 符号来自 12 个不同的 RS 码字。同理对于其他接入业务当$Q \ge 11$时，结合后续的分组交织可以使得目标数据流连续 12 个 RS 符号来自 12 个不同的 RS 码字。

在一种可能的实施方式中，选取$Q = 11$，卷积交织器具体结构如图 16（b）所示。其对应的交织时延约为 22*2*3/2=66 个 RS 符号。

采用如图 16（b）所示的卷积交织器分别对 32 条 PCS 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 5 到图 9 所示 PCS/FEC 通道数据流，不难理解，第一数据流最多包含G_{max} =8 个第一数据流子集，最少包含G_{min} =2 个第一数据流子集，为了对所有业务采用统一的交织方案，并降低卷积交织的延迟，第一数据流划分为$K1 = G_{max} = 8$个第一数据流分组，每个第一数据流分组内任意 2 条数据流来自相同的 RS 码字，对应分组具体划分为第一数据流 0-3 为第一数据流分组 0、第一数据流 4-7 为第一数据流分组 1、第一数据流 8-11 为第一数据流分组 2、第一数据流 12-15 为第一数据流分组 3、第一数据流 16-19 为第一数据流分组 4、第一数据流 20-23 为第一数据流分组 5、第一数据流 24-27 为第一数据流分组 6 和第一数据流 28-31 为第一数据流分组 7。从而本实施例采用图 35 所示的分组交织处理对应的参数为 K=K1=8, T=32/K1=4, 从每个第一数据流分组中各任取 1 条第一数据流作为分组交织 i（$0 \le i < 4$）的 8 条输入数据流。

图 38 为分组交织的一种实现方式示意图。如图 38 所示，分组交织 i（$0 \le i < 4$）的 8 个输入数据流分别为第一数据流 i，第一数据流 i+4，第一数据流 i+8，第一数据流 i+12，第一数据流 i+16，第一数据流 i+20，第一数据流 i+24，第一数据流 i+28；从每个输入数据流各获取连续的 B=6 个 RS 符号，构成 8 行 6 列第一符号矩阵，其中每行的 6 个 RS 符号分别为

图 16（b）所示卷积交织器的延迟线 0、延迟线 1 和延迟线 2 轮询 R=1 次输出的 6 个 RS 符号。结合图 5 到图 9 不难得出第一符号矩阵至少来自于 12 个不同 RS 码字，且最多 4 个 RS 符号属于同一个 RS 码字，进一步有第一符号矩阵第 0 行到第 3 行至少来自于 6 个不同 RS 码字，第 4 行到第 7 行至少来自于 6 个不同 RS 码字，且两者之间不来自于相同的 RS 码字。进一步有第一符号矩阵所有奇数列或者所有偶数列上 RS 码字分布可能是一致的。进一步有第一符号矩阵第 0 列到 3 列上 RS 码字分布可能是一致的，第 2 列到第 5 列上 RS 码字分布可能是一致的；因此可以从第一符号矩阵每列取 2 个 RS 符号，其中一个来自第 0 行到第 3 行中任意行，另外一个来自第 4 行到第 7 行中的任一行，共 12 个 RS 符号映射为第二符号矩阵的一行，得到 4 行 12 列的第二符号矩阵，第二符号矩阵第 t（$0 \le t < 4$）行的 12 个 RS 符号为分组交织 i 输出的目标数据流 i*4+t 中的连续 12 个 RS 符号，因此将第二符号矩阵一行定义为目标符号子集。为描述方便定义$S(i_x, j_y), i_x \in [0,7], j_y \in [0,5]$表示第一符号矩阵第$i_x$行第$j_y$列的符号。为使得目标符号子集的 12 个 RS 符号来自 12 个不同 RS 码字，则从第一符号矩阵的第 0 行到第 3 行取的 6 个 RS 符号或者从第 4 行到第 7 行取的 6 个 RS 符号对应于第一符号矩阵中$S(i_0, 0), S(i_1, 1), S(i_2, 2), S(i_3, 3), S(i_4, 4), S(i_5, 5)$其中$i_0, i_1, i_2, i_3$之间互不相等，$i_2, i_3, i_4, i_5$之间互不相等，$i_0, i_2, i_4$之间互不相等，$i_1, i_3, i_5$之间互不相等，$i_0, i_1, i_2, i_3, i_4, i_5 \in [0,3]$或者$i_0, i_1, i_2, i_3, i_4, i_5 \in [4,7]$，等效于目标符号子集中的每 2 个符号来自于第一符号矩阵同一列，且其中一个符号位于第一符号矩阵的第 0 行到第 3 行中的任意一行，另外一个符号位于第一符号矩阵的第 4 行到第 7 行中的任意一行，进一步目标符号子集中来自于第一符号矩阵的奇数列的 6 个符号位于第一符号矩阵的不同的行，来自于第一符号矩阵的偶数列的 6 个符号位于第一符号矩阵的不同的行；进一步目标符号子集中最少 1 个符号来自于第一符号矩阵的同一行，最多 2 个符号来自于第一符号矩阵的同一行，且来自于同一行的 2 个符号由卷积交织 2 个不同的延迟线输出，且此 2 个延迟线上对应延迟差值大于等于 2*d*Q=44 个符号。由此规则第一符号矩阵到第二符号矩阵的具体映射有很多种，表 1-表 4 给出了几种具体的映射关系，每个表第 y 行第 z 列的数字 x 表示第二符号矩阵第 y 行第 z 列的 RS 符号来自于第一符号矩阵的第 x 个 RS 符号，其中$0 \le y < 4, 0 \le z < 12, 0 \le x < 48$，第一符号矩阵的第 x 个 RS 符号表示第一符号矩阵第 x%8 行第$\lfloor x/8 \rfloor$列的符号。需要说明的是，将表 1-表 4 中进行任意行之间交换，或者任意列之间交换，或者任意行交换后在进行任意列交换，或者先任意列交换后在进行任意行交换都是一个合法的映射。应理解，x%8 表示 x 除以 8 之后的余数，$\lfloor x/8 \rfloor$表示 x 除以 8 之后的商，下面各实施例中不再对类似的表示方式进行赘述。

根据第一符号矩阵与第一数据流的关系，第一数据流和通道数据流关系，不难得出每个目标符号子集的 12 个符号来自 K1=8 个通道数据流，此 8 个通道数据流可以表示为通道数据流j_0,通道数据流j_1,通道数据流j_2,通道数据流j_3,通道数据流j_4,通道数据流j_5,通道数据流j_6,通道数据流j_7,且$\lfloor j_x/4 \rfloor = x, x \in [0,7], \lfloor j_x/4 \rfloor$表示对$j_x/4$向下取整；进一步从其中 4 个通道数据流各获取一个符号，从剩下的 4 个通道数流各获取 2 个 RS 符号；进一步当从某通道数据流获取 2 个 RS 符号，此 2 个 RS 符号在对应通道数据流间隔大于等于 a*N*K2/n，即大于等于 2*544*2/32=68 个 RS 符号；进一步最多 2 个 RS 符号来自 2 个不同通道数据流上对齐的 2 个符号。

表 1

0	4	9	13	18	22	27	31	33	37	40	44
1	5	8	12	19	23	26	30	32	36	41	45
2	6	11	15	16	20	25	29	35	39	42	46
3	7	10	14	17	21	24	28	34	38	43	47

表 2

0	4	9	13	19	23	26	30	33	37	40	44
1	5	8	12	18	22	27	31	32	36	41	45
2	6	11	15	17	21	24	28	35	39	42	46
3	7	10	14	16	20	25	29	34	38	43	47

表 3

0	4	9	13	19	23	26	30	33	37	40	44
1	5	8	12	18	22	27	31	32	36	41	45
2	6	11	15	17	21	24	28	35	39	42	46
3	7	10	14	16	20	25	29	34	38	43	47

表 4

0	4	9	13	19	23	26	30	33	37	40	44
1	5	8	12	18	22	27	31	32	36	41	45
2	6	11	15	16	20	25	29	35	39	42	46
3	7	10	14	17	21	24	28	34	38	43	47

经过分组交织生成 16 条目标数据流，对 16 条目标数据流中每个目标符号子集分别进行内码编码，其中内码编码信息位长度为 120 比特。具体地，内码编码器分别将目标数据流中每个目标符号子集总共 120 比特编码添加冗余，得到 16 条编码数据流。在一种可能的实施方式中，采用 Hamming(128,120)进行内码编码，分别将每条目标数据流中连续的 12 个 RS 符号总共 120 比特添加 8 比特冗余得到 128 比特的码字。在另一种可能的实施方式中，采用 BCH(136,120) 进行内码编码，分别将每条目标数据流中连续的 12 个 RS 符号总共 120 比特添加 16 比特冗余得到 136 比特的码字。

经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括调制映射（mapping），信道交织（channel interleaving），偏振划分（polarization distribution），或 DSP 成帧处理（framing）等。比如内码编码数据流可通过交织，以提升系统抗突发错误的能力。

采用本实施例的数据交织编码方案，其 KP4 RS(544,514) + Hamming(128,120)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 4.5E-3，性能逼近该级联 FEC 方案的最优性能。

实施例 2：对于所有业务采用统一交织，内码编码信息位长度为 120 比特，内码并行度 32。

采用如图 16（b）所示的卷积交织器分别对 32 条 PCS 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 5 到图 9 所示 PCS/FEC 通道数据流，不难理解，第一数据流最多包含G_{max} =8 个第一数据流子集，最少包含G_{min} =2 个第一数据流子集。为了对所有业务采用统

··的交织方案，并降低卷积交织的延迟，第··数据流划分为$K1 = G_{max} = 8$个第··数据流分组，每个第··数据流分组内任意 2 条数据流来自相同的 RS 码字，对应分组具体划分为第··数据流 0-3 为第··数据流分组 0、第··数据流为 4-7 第··数据流分组 1、第··数据流为 8-11 第··数据流分组 2、第··数据流 12-15 为第··数据流分组 3、第··数据流 16-19 为第··数据流分组 4、第··数据流 20-23 为第··数据流分组 5、第··数据流 24-27 为第··数据流分组 6 和第··数据流 28-31 为第··数据流分组 7。从而本实施例采用图 35 所示的分组交织处理对应的参数为 K=K1=8,T=32/K1=4,从每个第··数据流分组中各任取 1 条第··数据流作为分组交织 i（$0 \le i < 4$）的 8 条输入数据流。

图 39 为分组交织的另一种实现方式示意图。如图 39 所示，分组交织 i（$0 \le i < 4$）的 8 个输入数据流分别为第一数据流 i，第一数据流 i+4，第一数据流 i+8，第一数据流 i+12，第一数据流 i+16，第一数据流 i+20，第一数据流 i+24，第一数据流 i+28。从每个输入数据流各获取连续的 12 个 RS 符号，构成 8 行 12 列第一符号矩阵，其中每行的 12 个 RS 符号分别为图 16（b）所示卷积交织器的延迟线 0、延迟线 1 和延迟线 2 轮询 R=2 次连续输出的 12 个 RS 符号。结合图 5 到图 9 不难得出第一符号矩阵至少来自于 12 个不同 RS 码字，且最多 R*K1/K2=8 个 RS 符号属于同一个 RS 码字，进一步有第一符号矩阵第 0 行到第 3 行至少来自于 6 个不同 RS 码字，第 4 行到第 7 行至少来自于 6 个不同 RS 码字，且两者之间不来自于相同的 RS 码字，进一步有第一符号矩阵所有奇数列或者所有偶数列上 RS 码字分布是可能一致的；进一步有第一符号矩阵第 0 列到第 3 列的各列 RS 码字分布可能是一致的，第 2 列到第 5 列的各列上 RS 码字分布可能是一致的，第 6 列到第 9 列的各列上 RS 码字分布可能是一致的，第 8 列到第 11 列的各列上 RS 码字分布可能是一致的；进一步有第一符号矩阵的第 j 列、第 j+1、第 j+6 列和第 j+7 列（$j \in \{0,2,4\}$）的 RS 码字分布可能是一致的；因此可以从第一符号矩阵的每列各取一个符号共 12 个 RS 符号映射为第二符号矩阵的一行，以得到 8 行 12 列的第二符号矩阵，第二符号矩阵第 t（$0 \le t < 8$）行的 12 个 RS 符号为分组交织 i 输出的目标数据流 i*8+t 中的连续 12 个 RS 符号，因此将第二符号矩阵一行定义为目标符号子集。为描述方便定义 $S(i_x, j_y), i_x \in [0,7], j_y \in [0,11]$表示第一符号矩阵第$i_x$行第$j_y$列的符号。为使得目标符号子集的 12 个 RS 符号来自 12 个不同 RS 码字，此 12 个 RS 符号对应于第一符号矩阵中 $S(i_0,0), S(i_1,1), S(i_2,2) \cdots S(i_{10},10), S(i_{11},11)$，其中$i_0, i_1, i_2, i_3$之间互不相等，$i_2, i_3, i_4, i_5$之间互不相等，$i_6, i_7, i_8, i_9$之间互不相等，$i_8, i_9, i_{10}, i_{11}$之间互不相等，$i_0, i_1, i_6, i_7$互不相等，$i_2, i_3, i_8, i_9$互不相等，$i_4, i_5, i_{11}, i_{12}$互不相等，$i_0, i_2, i_4, i_6, i_8, i_{10}$之间互不相等，$i_1, i_3, i_5, i_7, i_9, i_{11}$之间互不相等，$i_0, i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10}, i_{11} \in [0,7]$，此等效于目标符号子集中的每 1 个符号来自于第一符号矩阵同一列；进一步目标符号子集中来自于第一符号矩阵的奇数列的 6 个符号位于第一符号矩阵的不同的行，来自于第一符号矩阵的偶数列的 6 个符号位于第一符号矩阵的不同的行；进一步目标符号子集来自于第一符号矩阵由延迟取值相同延迟线输出的符号位于第一符号矩阵不同的行；进一步目标符号子集中最少 1 个符号来自于第一符号矩阵的同一行，最多 2 个符号来自于第一符号矩阵的同一行，且来自于同一行的 2 个符号由卷积交织 2 个不同的延迟线输出，且此 2 个延迟线对应延迟差值大于等于 2*d*Q=44 个符号。由此规则第一符号矩阵到第二符号矩阵的具体映射有很多种，下面表 5 到表 8 给出了几种具体的映射关系，每

个表第 y 行第 z 列的数字 x 表示第二符号矩阵第 y 行第 z 列的 RS 符号来自于第一符号矩阵的第 x 个 RS 符号，其中$0 \le y < 8, 0 \le z < 12, 0 \le x < 96$，第一符号矩阵的第 x 个 RS 符号表示第一符号矩阵第 x%8 行第$\lfloor x/8 \rfloor$列的符号。需要说明的是将表中进行任意行之间交换，或者任意列之间交换，或者任意行交换后在进行任意列交换，或者先任意列交换后在进行任意行交换都是一个合法的映射。

根据第一符号矩阵中 RS 码字分布规律，第一符号矩阵到目标符号子集的另外一种映射方式为：从第一符号矩阵第 2*j 列和第 2*j+1 列($j \in [0,5]$)中的任取一列总共 6 列，再从选取的 6 列中的各列取 2 个 RS 符号共 12 个 RS 符号映射为一个目标符号子集，对应取得的 12 个 RS 对应于第一符号矩阵的 $S(i_0,j_0), S(i_1,j_0), S(i_2,j_1)$，$S(i_3,j_1), S(i_4,j_2), S(i_5,j_2)$，$S(i_6,j_3), S(i_7,j_3), S(i_8,j_4), S(i_9,j_4), S(i_{10},j_5), S(i_{11},j_5)$；$i_0, i_1, i_2, i_3$互不相等，$i_2, i_3, i_4, i_5$互不相等，$i_6, i_7, i_8, i_9$互不相等，$i_8, i_9, i_{10}, i_{11}$互不相等，$i_0, i_1, i_6, i_7$互不相等，$i_2, i_3, i_8, i_9$互不相等，$i_4, i_5, i_{10}, i_{11}$互不相等，$i_0, i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10}, i_{11}, i_{12}, i_{13}, i_{14}, i_{15} \in [0,7]$，$j_0 \in [0,1], j_1 \in [2,3], j_2 \in [4,5], j_3 \in [6,7], j_4 \in [8,9], j_5 \in [10,11], j_6 \in [12,13], j_7 \in [14,15]$；等效于目标符号子集中的每 2 个符号来自于第一符号矩阵同一列，且其中一个符号位于第一符号矩阵的第 0 行到第 3 行中的任意一行，另外一个符号位于第一符号矩阵的第 4 行到第 7 行中的任意一行；进一步目标符号子集中来自于第一符号矩阵的奇数列的 6 符号位于第一符号矩阵的不同的行，来自于第一符号矩阵的偶数列的 6 个符号位于第一符号矩阵的不同的行；进一步目标符号子集来自于第一符号矩阵由延迟取值相同延迟线输出的符号位于第一符号矩阵不同的行；进一步目标符号子集中最多 2 个符号来自于第一符号矩阵的同一行，且来自于同一行的 2 个符号由卷积交织 2 个不同的延迟线输出，且此 2 个延迟线对应延迟差值大于等于 2*d*Q=44 个符号。由此规则第一符号矩阵到第二符号矩阵的具体映射有很多种，表 9 到表 12 给出了几种具体的映射关系，每个表第 y 行第 z 列的数字 x 表示第二符号矩阵第 y 行第 z 列的 RS 符号来自于第一符号矩阵的第 x 个 RS 符号，其中$0 \le y < 8, 0 \le z < 12, 0 \le x < 96$，第一符号矩阵的第 x 个 RS 符号表示第一符号矩阵第 x%8 行第$\lfloor x/8 \rfloor$列的符号。需要说明的是将表中进行任意行之间交换，或者任意列之间交换，或者任意行交换后在进行任意列交换，或者先任意列交换后在进行任意行交换都是一个合法的映射。

根据第一符号矩阵与第一数据流的关系，第一数据流和通道数据流关系，不难得出每个目标符号子集的 12 个符号来自 K1=8 个通道数据流，此 8 个通道数据流可以表示为通道数据流j_0,通道数据流j_1,通道数据流j_2,通道数据流j_3,通道数据流j_4,通道数据流j_5,通道数据流j_6,通道数据流j_7,且$\lfloor j_x/4 \rfloor = x, x \in [0,7]$,$\lfloor j_x/4 \rfloor$表示对$j_x/4$向下取整；进一步从其中 4 个通道数据流各获取一个符号，从剩下的 4 个通道数流各获取 2 个 RS 符号；进一步当从某通道数据流获取 2 个 RS 符号，此 2 个 RS 符号在对应通道数据流间隔大于等于 a*N*K2/n，即大于等于 2*544*2/32=68 个 RS 符号；进一步最多 2 个 RS 符号来自 2 个不同通道数据流上对齐的 2 个 RS 符号。

表 5

0	9	18	27	33	40	52	61	70	79	85	92
1	8	19	26	32	41	53	60	71	78	84	93
2	11	16	25	35	42	54	63	68	77	87	94
3	10	17	24	34	43	55	62	69	76	86	95
4	13	22	31	37	44	48	57	66	75	81	88
5	12	23	30	36	45	49	56	67	74	80	89
6	15	20	29	39	46	50	59	64	73	83	90
7	14	21	28	38	47	51	58	65	72	82	91

表 6

0	9	18	27	33	40	52	61	70	79	85	92
1	8	19	26	32	41	53	60	71	78	84	93
2	11	17	24	35	42	54	63	69	76	87	94
3	10	16	25	34	43	55	62	68	77	86	95
4	13	22	31	37	44	48	57	66	75	81	88
5	12	23	30	36	45	49	56	67	74	80	89
6	15	21	28	39	46	50	59	65	72	83	90
7	14	20	29	38	47	51	58	64	73	82	91

表 7

0	9	19	26	33	40	52	61	71	78	85	92
1	8	18	27	32	41	53	60	70	79	84	93
2	11	17	24	35	42	54	63	69	76	87	94
3	10	16	25	34	43	55	62	68	77	86	95
4	13	23	30	37	44	48	57	67	74	81	88
5	12	22	31	36	45	49	56	66	75	80	89
6	15	21	28	39	46	50	59	65	72	83	90
7	14	20	29	38	47	51	58	64	73	82	91

表 8

0	9	19	26	33	40	52	61	71	78	85	92
1	8	18	27	32	41	53	60	70	79	84	93
2	11	16	25	35	42	54	63	68	77	87	94
3	10	17	24	34	43	55	62	69	76	86	95
4	13	23	30	37	44	48	57	67	74	81	88
5	12	22	31	36	45	49	56	66	75	80	89
6	15	20	29	39	46	50	59	64	73	83	90
7	14	21	28	38	47	51	58	65	72	82	91

表 9

0	4	18	22	33	37	57	61	75	79	88	92
1	5	19	23	32	36	56	60	74	78	89	93
2	6	16	20	35	39	59	63	73	77	90	94
3	7	17	21	34	38	58	62	72	76	91	95
9	13	27	31	40	44	48	52	66	70	81	85
8	12	26	30	41	45	49	53	67	71	80	84
11	15	25	29	42	46	50	54	64	68	83	87
10	14	24	28	43	47	51	55	65	69	82	86

表 10

0	4	18	22	33	37	57	61	75	79	88	92
1	5	19	23	32	36	56	60	74	78	89	93
2	6	17	21	35	39	59	63	72	76	90	94
3	7	16	20	34	38	58	62	73	77	91	95
9	13	27	31	40	44	48	52	66	70	81	85
8	12	26	30	41	45	49	53	67	71	80	84
11	15	24	28	42	46	50	54	65	69	83	87
10	14	25	29	43	47	51	55	64	68	82	86

表 11

0	4	19	23	33	37	57	61	74	78	88	92
1	5	18	22	32	36	56	60	75	79	89	93
2	6	17	21	35	39	59	63	72	76	90	94
3	7	16	20	34	38	58	62	73	77	91	95
9	13	26	30	40	44	48	52	67	71	81	85
8	12	27	31	41	45	49	53	66	70	80	84
11	15	24	28	42	46	50	54	65	69	83	87
10	14	25	29	43	47	51	55	64	68	82	86

表 12

0	4	19	23	33	37	57	61	74	78	88	92
1	5	18	22	32	36	56	60	75	79	89	93
2	6	16	20	35	39	59	63	73	77	90	94
3	7	17	21	34	38	58	62	72	76	91	95
9	13	26	30	40	44	48	52	67	71	81	85
8	12	27	31	41	45	49	53	66	70	80	84
11	15	25	29	42	46	50	54	64	68	83	87
10	14	24	28	43	47	51	55	65	69	82	86

经过分组交织生成 32 条目标数据流，对 32 条目标数据流中每个目标符号子集分别进行内码编码，其中内码编码信息位长度为 120 比特。具体地，内码编码器分别将目标数据流中每个目标符号子集总共 120 比特编码添加冗余，得到 32 条编码数据流。在一种可能的实施方式中，采用 Hamming(128,120)进行内码编码，分别将每条目标数据流中连续的 12 个 RS 符号总共 120 比特添加 8 比特冗余得到 128 比特的码字。在另一种可能的实施方式中，采用

BCH(136,120) 进行内码编码，分别将每条目标数据流中连续的 12 个 RS 符号总共 120 比特添加 16 比特冗余得到 136 比特的码字。

经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括调制映射（mapping），信道交织（channel interleaving），偏振划分（polarization distribution），或 DSP 成帧处理（framing）等。比如内码编码数据流可通过交织，以提升系统抗突发错误的能力。

采用本实施例的数据交织编码方案，其 KP4 RS(544,514) + Hamming(128,120)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 4.5E-3，性能逼近该级联 FEC 方案的最优性能。

实施例 3：对于所有业务采用统一交织，内码编码信息位长度为 160 比特，内码并行度 16。

在本实施例中，卷积交织处理采用如图 11 所示的结构，分别对 n=32 条 PCS 通道数据流进行卷积交织得到 n=32 条第一数据流。其中卷积交织 0、卷积交织 1、卷积交织 2、…、卷积交织 31 采用相同的交织结构。图 18（a）给出了一种卷积交织器结构，其包含 p=4 条延迟线（delay line）。这 4 条延迟线分别包括 3Q 个存储单元、2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=2 个符号。也就是，延迟线 0 的延迟取值为 6Q 个符号，延迟线 1 的延迟取值为 4Q 个符号，延迟线 2 的延迟取值为 2Q 个符号，延迟线 3 的延迟取值为 0 个符号即无延迟。

如图 18(a)所示的 $C_r()$ 表示该通道数据流$r(0 \le r \le n-1)$中一个 RS 符号。比如，$C_r(8t)$，$C_r(8t+1)$表示通道数据流中当前输入给延迟线 0 的两个 RS 符号且$C_r(8t-24Q)$，$C_r(8t-24Q+1)$为延迟线 0 输出的两个 RS 符号；$C_r(8t+2)$，$C_r(8t+3)$表示通道数据流中接下来输入给延迟线 1 的两个 RS 符号且$C_r(8t-16Q+2)$，$C_r(8t-16Q+3)$为延迟线 1 输出的两个 RS 符号；$C_r(8t+4)$，$C_r(8t+5)$表示通道数据流中再接下来输入给延迟线 2 的两个 RS 符号且$C_r(8t-8Q+4)$，$C_r(8t-8Q+5)$为延迟线 2 输出的两个 RS 符号；$C_r(8t+6)$，$C_r(8t+7)$表示通道数据流中再随后输入给延迟线 3 的两个 RS 符号且$C_r(8t+6)$，$C_r(8t+7)$为延迟线 3 输出的两个 RS 符号，依次类推。结合图 5 和图 6，可以看出当$8Q+2 \ge 68$即$Q \ge 9$时，卷积交织输出的$C_r(8t-24Q)$，$C_r(8t-24Q+1)$，$C_r(8t-16Q+2)$，$C_r(8t-16Q+3)$，$C_r(8t-8Q+4)$，$C_r(8t-8Q+5)$，$C_r(8t+6)$，$C_r(8t+7)$总共 8 个 RS 符号来自 8 个不同的 RS 码字。再利用分组交织将 2 个第一数据流子集的数据汇聚可以使得目标数据流连续 16 个 RS 符号来自 16 个不同的 RS 码字。同理对于其他接入业务当$Q \ge 9$时，结合后续的分组交织可以使得目标数据流连续 16 个 RS 符号来自 16 个不同的 RS 码字。

在一种可能的实施方式中，选取$Q = 9$，卷积交织器具体结构如图 18（b）所示。其对应的交织时延约为 27*2*4/2=108 个 RS 符号。

采用如图 18（b）所示的卷积交织器分别对 32 条 PCS 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 5 到图 9 所示 PCS/FEC 通道数据流，不难理解，第一数据流最多包

含G_{max}=8 个第一数据流子集，最少包含G_{min}=2 个第一数据流子集。为了对所有业务采用统一的交织方案，并降低卷积交织的延迟，第一数据流划分为$K1 = G_{max} = 8$个第一数据流分组，每个第一数据流分组内任意 2 条数据流来自相同的 RS 码字，对应分组具体划分为第一数据流 0-3 为第一数据流分组 0、第一数据流 4-7 为第一数据流分组 1、第一数据流 8-11 为第一数据流分组 2、第一数据流 12-15 为第一数据流分组 3、第一数据流 16-19 为第一数据流分组 4、第一数据流 20-23 为第一数据流分组 5、第一数据流 24-27 为第一数据流分组 6 和第一数据流 28-31 为第一数据流分组 7。从而本实施例采用图 35 所示的分组交织处理对应的参数为 K=K1=8, T=32/K1=4, 从每个第一数据流分组中各任取 1 条第一数据流作为分组交织 i（$0 \le i < 4$）的 8 条输入数据流。

图 40 为分组交织的另一种实现方式示意图。如图 40 所示，分组交织 i（$0 \le i < 4$）的 8 个输入数据流分别为第一数据流 i，第一数据流 i+4，第一数据流 i+8，第一数据流 i+12，第一数据流 i+16，第一数据流 i+20，第一数据流 i+24，第一数据流 i+28；从每个输入数据流各获取连续的 8 个 RS 符号，构成 8 行 8 列第一符号矩阵，其中每行的 8 个 RS 符号分别为图 18（b）所示卷积交织器的延迟线 0、延迟线 1、延迟线 2 和延迟线 3 轮询 R=1 次输出的 8 个 RS 符号。结合图 5 到图 9 不难得出第一符号矩阵来自于 16 个不同的 RS 码字且每 4 个 RS 符号属于同一个码字，进一步有第一符号矩阵第 0 行到第 3 行来自于 8 个不同 RS 码字，第 4 行到第 7 行来自于 8 个不同 RS 码字，且两者之间不来自于相同的 RS 码字，进一步有第一符号矩阵所有奇数列或者所有偶数列上 RS 码字分布可能是一致的；进一步有第一符号矩阵能第 0 列到 3 列上 RS 码字分布可能是一致的，第 2 列到第 5 列上 RS 码字分布可能是一致的，第 4 列到第 7 列上 RS 码字分布可能是一致的；因此可以从第一符号矩阵每列取 2 个 RS 符号，其中一个来自第 0 行到第 3 行中任意行，另外一个来自第 4 行到第 7 行中的任一行，共 16 个 RS 符号映射为第二符号矩阵的一行，得到 4 行 16 列的第二符号矩阵，第二符号矩阵第 t（$0 \le t < 4$）行的 16 个 RS 符号为分组交织 i 输出的目标数据流 i*4+t 中的连续 16 个 RS 符号，因此将第二符号矩阵一行定义为目标符号子集。为描述方便定义$S(i_x, j_y), i_x \in [0,7], j_y \in [0,7]$表示第一符号矩阵第$i_x$行第$j_y$列的符号。为使得目标符号子集的 16 个 RS 符号来自 16 个不同 RS 码字，则从第一符号矩阵的第 0 行到第 3 行取的 8 个 RS 符号或者从第 4 行到第 7 行取的 8 个 RS 符号对应于第一符号矩阵中

$S(i_0, 0), S(i_1, 1), S(i_2, 2), S(i_3, 3), S(i_4, 4), S(i_5, 5), S(i_6, 6), S(i_7, 7)$其中$i_0, i_1, i_2, i_3$之间互不相等，$i_2, i_3, i_4, i_5$之间互不相等，$i_4, i_5, i_6, i_7$之间互不相等，$i_0, i_2, i_4, i_6$之间互不相等，$i_1, i_3, i_5, i_7$之间互不相等，$i_0, i_1, i_2, i_3, i_4, i_5, i_6, i_7 \in [0,3]$或者$i_0, i_1, i_2, i_3, i_4, i_5, i_6, i_7 \in [4,7]$，等效于目标符号子集中的每 2 个符号来自于第一符号矩阵的同一列，且其中一个符号位于第一符号矩阵的第 0 行到第 3 行中的任意一行，另外一个符号位于第一符号矩阵的第 4 行到第 7 行中的任意一行；进一步目标符号子集中来自于第一符号矩阵的奇数列的 8 个符号位于第一符号矩阵的不同的行，来自于第一符号矩阵的偶数列的 8 个符号位于第一符号矩阵的不同的行；进一步目标符号子集中每 2 个符号来自于第一符号矩阵的同一行，且来自于同一行的 2 个符号由卷积交织 2 个不同的延迟线输出，且此 2 个延迟线对应延迟差值大于等于 2*d*Q=44 个符号。由此规则第一符号矩阵到第二符号矩阵的具体映射有很多种，下面表 13 到表 16 给出了几种具体的映射关系，每个表第 y 行第 z 列的数字 x 表示第二符号矩阵第 y 行第 z 列的 RS 符号来自于第一符号矩阵的第 x 个 RS 符号，其中$0 \le y < 4, 0 \le z < 16, 0 \le x < 64$，第一符号矩阵的第 x 个 RS 符号表示第一符号矩阵第 x%8 行第$\lfloor x/8 \rfloor$列的符号。需要说明的是将表进行任意行之间交换，或者任意列之间交换，或者任意行交换后在进行任意列交换，或者先任意列交换后在

进行任意行交换都是一个合法的映射。

根据第一符号矩阵与第一数据流的关系，第一数据流和通道数据流关系，不难得出每个目标符号子集的 16 个符号来自 K1=8 个通道数据流，此 8 个通道数据流可以表示为通道数据流j_0,通道数据流j_1,通道数据流j_2,通道数据流j_3,通道数据流j_4,通道数据流j_5,通道数据流j_6,通道数据流j_7,且$\lfloor j_x/4 \rfloor = x, x \in [0,7]$,$\lfloor j_x/4 \rfloor$表示对$j_x/4$向下取整；进一步从每个通道数据流各获取 2 个符号，此 2 个 RS 符号在对应通道数据流间隔大于等于 a*N*K2/n，即大于等于 2*544*2/32=68 个 RS 符号；进一步最多 2 个 RS 符号来自 2 个不同通道数据流上对其的 2 个 RS 符号。

表 13

0	4	9	13	18	22	27	31	33	37	40	44	51	55	58	62
1	5	8	12	19	23	26	30	32	36	41	45	50	54	59	63
2	6	11	15	16	20	25	29	35	39	42	46	49	53	56	60
3	7	10	14	17	21	24	28	34	38	43	47	48	52	57	61

表 14

0	4	9	13	18	22	27	31	33	37	40	44	51	55	58	62
1	5	8	12	19	23	26	30	32	36	41	45	50	54	59	63
2	6	11	15	17	21	24	28	35	39	42	46	48	52	57	61
3	7	10	14	16	20	25	29	34	38	43	47	49	53	56	60

表 15

0	4	9	13	19	23	26	30	33	37	40	44	50	54	59	63
1	5	8	12	18	22	27	31	32	36	41	45	51	55	58	62
2	6	11	15	17	21	24	28	35	39	42	46	48	52	57	61
3	7	10	14	16	20	25	29	34	38	43	47	49	53	56	60

表 16

0	4	9	13	19	23	26	30	33	37	40	44	50	54	59	63
1	5	8	12	18	22	27	31	32	36	41	45	51	55	58	62
2	6	11	15	16	20	25	29	35	39	42	46	49	53	56	60
3	7	10	14	17	21	24	28	34	38	43	47	48	52	57	61

经过分组交织生成 16 条目标数据流，对 16 条目标数据流中每个目标符号子集分别进行内码编码，其中内码编码信息位长度为 160 比特。具体地，内码编码器分别将目标数据流中每个目标符号子集总共 160 比特编码添加冗余，得到 16 条编码数据流。在一种可能的实施方式中，采用 Hamming(170,160)进行内码编码，分别将每条目标数据流中连续的 16 个 RS 符号总共 160 比特添加 10 比特冗余得到 170 比特的码字。在另一种可能的实施方式中，采用 BCH(176,160) 进行内码编码，分别将每条目标数据流中连续的 16 个 RS 符号总共 160 比特添加 16 比特冗余得到 176 比特的码字。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。

经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括调制映射（mapping），信道交织（channel interleaving），偏振划分（polarization distribution），或 DSP 成帧处理（framing）等。比如内码编码数据流可通过交织，以提升系统抗突发错误的能力。

采用本实施例的数据交织编码方案，当内码采用 Hamming(170,160)时，其 KP4 RS(544,514) + Hamming(170,160)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 4.3E-3，性能逼近该级联 FEC 方案的最优性能。当内码采用 BCH(176,160)时，其 KP4 RS(544,514) + BCH(176,160)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 8.3E-3，性能逼近该级联 FEC 方案的最优性能。

实施例 4：对于所有业务采用统一交织，内码编码信息位长度为 160 比特，内码并行度 32。

采用如图 18（b）所示的卷积交织器分别对 32 条 PCS 通道数据流进行卷积交织处理得到 32 条第一数据流。参考图 5 到图 9 所示 PCS/FEC 通道数据流，不难理解，第一数据流最多包含G_{max} =8 个第一数据流子集，最少包含G_{min} =2 个第一数据流子集。为了对所有业务采用统一的交织方案，并降低卷积交织的延迟，第一数据流划分为$K1 = G_{max} = 8$个第一数据流分组，每个第一数据流分组内任意 2 条数据流来自相同的 RS 码字，对应分组具体划分为第一数据流 0-3 为第一数据流分组 0、第一数据流 4-7 为第一数据流分组 1、第一数据流 8-11 为第一数据流分组 2、第一数据流 12-15 为第一数据流分组 3、第一数据流 16-19 为第一数据流分组 4、第一数据流 20-23 为第一数据流分组 5、第一数据流 24-27 为第一数据流分组 6 和第一数据流 28-31 为第一数据流分组 7。从而本实施例采用图 35 所示的分组交织处理对应的参数为 K=K1=8,T=32/K1=4 从每个第一数据流分组中各任取 1 条第一数据流作为分组交织 i（$0 \le i < 4$）的 8 条输入数据流。

图 41 为分组交织的另一种实现方式示意图。如图 41 所示，分组交织 i（$0 \le i < 4$）的 8 个输入数据流分别为第一数据流 i，第一数据流 i+4，第一数据流 i+8，第一数据流 i+12，第一数据流 i+16，第一数据流 i+20，第一数据流 i+24，第一数据流 i+28；从每个输入数据流各获取连续的 16 个 RS 符号，构成 8 行 16 列第一符号矩阵，其中每行的 16 个 RS 符号分别为图 18（b）所示卷积交织器的延迟线 0、延迟线 1、延迟线 2 和延迟线 3 轮询 2 次连续输出的 16 个 RS 符号。结合图 5 到图 9 不难得出第一符号矩阵来自于 16 个不同 RS 码字，且每 8 个 RS 符号属于同一个 RS 码字，进一步有第一符号矩阵第 0 行到第 3 行至少来自于 8 个不同 RS 码字，第 4 行到第 7 行至少来自于 8 个不同 RS 码字，且两者之间不来自于相同的 RS 码字，进一步有第一符号矩阵所有奇数列或者所有偶数列上 RS 码字分布可能是一致的；进一步有第一符号矩阵第 0 列到第 3 列的各列上 RS 码字分布可能是一致的，第 2 列到第 5 列的各列上 RS 码字分布可能是一致的，第 4 列到第 7 列的各列上 RS 码字分布可能是一致的，第一符号矩阵有可能第 8 列第 11 列的各列上 RS 码字分布可能是一致的，第 10 列到第 13 列的各列上 RS 码字分布可能是一致的，第 12 列到第 15 列的各列上 RS 码字分布可能是一致的；进一步有第一符号矩阵有可能第 j 列，第 j+1 列，第 j+8 列和第 j+9 列（$j \in \{0,2,4,6\}$）的各列的 RS 码字分布是一致的；因此可以从第一符号矩阵的每列各取一个符号共 16 个 RS 符号映射为第二符号矩阵的一行，以得到 8 行 16 列的第二符号矩阵，第二符号矩阵第 t（$0 \le t < 8$）行的 16 个 RS 符号为分组交织 i 输出的目标数据流 i*8+t 中的连续 16 个 RS 符号，因此将第二符号矩阵一行定义为目标符号子集。为描述方便定义$S(i_x, j_y), i_x \in [0,7], j_y \in [0,15]$表示第一符号矩阵第$i_x$行第$j_y$列的符号。为使得目标符号子集的 16 个 RS 符号来自 16 个不同 RS 码字，此 16 个 RS 符号对应于第一符号矩阵中$S(i_0,0), S(i_1,1), S(i_2,2) \ldots S(i_{14},14), S(i_{15},15)$，其中$i_0, i_1, i_2, i_3$之间互不相等，$i_2, i_3, i_4, i_5$之间互不相等，$i_4, i_5, i_6, i_7$之间互不相等，$i_8, i_9, i_{10}, i_{11}$之间互不相等，

$i_{10},i_{11},i_{12},i_{13}$之间互不相等，$i_{12},i_{13},i_{14},i_{15}$之间互不相等，$i_0,i_1,i_8,i_9$互不相等，$i_2,i_3,i_{10},i_{11}$互不相等，$i_4,i_5,i_{12},i_{13}$互不相等，$i_6,i_7,i_{14},i_{15}$互不相等，$i_0,i_2,i_4,i_6,i_8,i_{10},i_{12},i_{14}$之间互不相等，$i_1,i_3,i_5,i_7,i_9,i_{11},i_{13},i_{15}$之间互不相等，$i_0,i_1,i_2,i_3,i_4,i_5,i_6,i_7,i_8,i_9,i_{10},i_{11},i_{12},i_{13},i_{14},i_{15} \in [0,7]$；等效于目标符号子集中的每 1 个符号来自于第一符号矩阵同一列，进一步目标符号子集中来自于第一符号矩阵的奇数列的 8 个符号位于第一符号矩阵的不同的行，来自于第一符号矩阵的偶数列的 8 个符号位于第一符号矩阵的不同的行；进一步目标符号子集来自于第一符号矩阵由延迟取值相同延迟线输出的符号位于第一符号矩阵不同的行；进一步目标符号子集中每 2 个符号来自于第一符号矩阵的同一行，且来自于同一行的 2 个符号由卷积交织 2 个不同的延迟线输出，且此 2 个延迟线对应延迟差值大于等于 2*d*Q=44 个符号。由此规则第一符号矩阵到第二符号矩阵的具体映射有很多种，下面表 17 到表 20 给出了几种具体的映射关系，每个表第 y 行第 z 列的数字 x 表示第二符号矩阵第 y 行第 z 列的 RS 符号来自于第一符号矩阵的第 x 个 RS 符号，其中$0 \le y < 8, 0 \le z < 16, 0 \le x < 128$，第一符号矩阵的第 x 个 RS 符号表示第一符号矩阵第 x%8 行第$\lfloor x/8 \rfloor$列的符号。需要说明的是将表中进行任意行之间交换，或者任意列之间交织，或者任意行交织后在进行任意列交换，或者先任意列交换后在进行任意行交换都是一个合法的映射。

根据第一符号矩阵中 RS 码字分布规律，第一符号矩阵到目标符号子集的另外一种映射方式为：从第一符号矩阵第 2*j 列和第 2*j+1 列($j \in [0,7]$)中的任取一列总共 8 列，再从选取的 8 列中的各列取 2 个 RS 符号共 16 个 RS 符号映射为一个目标符号子集，对应取得的 16 个 RS 对应于第一符号矩阵的 $S(i_0,j_0),S(i_1,j_0),S(i_2,j_1)$，$S(i_3,j_1),S(i_4,j_2),S(i_5,j_2)$，$S(i_6,j_3),S(i_7,j_3),S(i_8,j_4),S(i_9,j_4),S(i_{10},j_5),S(i_{11},j_5),S(i_{12},j_6),S(i_{13},j_6),S(i_{14},j_7),S(i_{15},j_7)$；$i_0,i_1,i_2,i_3$互不相等，$i_2,i_3,i_4,i_5$互不相等，$i_4,i_5,i_6,i_7$互不相等，$i_8,i_9,i_{10},i_{11}$互不相等，$i_{10},i_{11},i_{12},i_{13}$互不相等，$i_{12},i_{13},i_{14},i_{15}$互不相等，$i_0,i_1,i_8,i_9$互不相等，$i_2,i_3,i_{10},i_{11}$互不相等，$i_4,i_5,i_{12},i_{13}$互不相等，$i_6,i_7,i_{14},i_{15}$互不相等，$i_0,i_1,i_2,i_3,i_4,i_5,i_6,i_7,i_8,i_9,i_{10},i_{11},i_{12},i_{13},i_{14},i_{15} \in [0,7]$，$j_0 \in [0,1], j_1 \in [2,3], j_2 \in [4,5], j_3 \in [6,7], j_4 \in [8,9], j_5 \in [10,11], j_6 \in [12,13], j_7 \in [14,15]$；等效于目标符号子集中的每 2 个符号来自于第一符号矩阵同一列，且其中一个符号位于第一符号矩阵的第 0 行到第 3 行中的任意一行，另外一个符号位于第一符号矩阵的第 4 行到第 7 行中的任意一行；进一步目标符号子集中来自于第一符号矩阵的奇数列的 8 个符号位于第一符号矩阵的不同的行，来自于第一符号矩阵的偶数列的 8 个符号位于第一符号矩阵的不同的行；进一步目标符号子集来自于第一符号矩阵由延迟取值相同延迟线输出的符号位于第一符号矩阵不同的行；进一步目标符号子集中每 2 个符号来自于第一符号矩阵的同一行，且来自于同一行的 2 个符号由卷积交织 2 个不同的延迟线输出，且此 2 个延迟线对应延迟差值大于等于 2*d*Q=44 个符号。由此规则第一符号矩阵到第二符号矩阵的具体映射有很多种，表 21 到表 24 给出了几种具体的映射关系，每个表第 y 行第 z 列的数字 x 表示第二符号矩阵第 y 行第 z 列的 RS 符号来自于第一符号矩阵的第 x 个 RS 符号，其中$0 \le y < 8, 0 \le z < 16, 0 \le x < 128$，第一符号矩阵的第 x 个 RS 符号表示第一符号矩阵第 x%8 行第$\lfloor x/8 \rfloor$列的符号。需要说明的是将表中进行任意行之间交换，或者任意列之间交织，或者任意行交织后在进行任意列交换，或者先任意列交换后在进行任意行交换都是一个合法的映射。

根据第一符号矩阵与第一数据流的关系，第一数据流和通道数据流关系，不难得出每个目标符号子集的 16 个符号来自 K1=8 个通道数据流，此 8 个通道数据流可以表示为通道数据

流j_0,通道数据流j_1,通道数据流j_2,通道数据流j_3,通道数据流j_4,通道数据流j_5,通道数据流j_6,通道数据流j_7,且$\lfloor j_x/4 \rfloor = x, x \in [0,7]$,$\lfloor j_x/4 \rfloor$表示对$j_x/4$向下取整;进一步从每个通道数据流各获取2个符号,此2个RS符号在对应通道数据流间隔大于等于a*N*K2/n,即大于等于2*544*2/32=68个RS符号;进一步最多2个RS符号来自2个不同通道数据流上对齐的2个RS符号。

表17

0	9	18	27	33	40	51	58	68	77	86	95	101	108	119	126
2	11	16	25	35	42	49	56	70	79	84	93	103	110	117	124
1	8	19	26	32	41	50	59	69	76	87	94	100	109	118	127
3	10	17	24	34	43	48	57	71	78	85	92	102	111	116	125
4	13	22	31	37	44	55	62	64	73	82	91	97	104	115	122
6	15	20	29	39	46	53	60	66	75	80	89	99	106	113	120
5	12	23	30	36	45	54	63	65	72	83	90	96	105	114	123
7	14	21	28	38	47	52	61	67	74	81	88	98	107	112	121

表18

0	9	18	27	33	40	51	58	68	77	86	95	101	108	119	126
2	11	17	24	35	42	48	57	70	79	85	92	103	110	116	125
1	8	19	26	32	41	50	59	69	76	87	94	100	109	118	127
3	10	16	25	34	43	49	56	71	78	84	93	102	111	117	124
4	13	22	31	37	44	55	62	64	73	82	91	97	104	115	122
6	15	21	28	39	46	52	61	66	75	81	88	99	106	112	121
5	12	23	30	36	45	54	63	65	72	83	90	96	105	114	123
7	14	20	29	38	47	53	60	67	74	80	89	98	107	113	120

表19

0	9	19	26	33	40	50	59	68	77	87	94	101	108	118	127
2	11	17	24	35	42	48	57	70	79	85	92	103	110	116	125
1	8	18	27	32	41	51	58	69	76	86	95	100	109	119	126
3	10	16	25	34	43	49	56	71	78	84	93	102	111	117	124
4	13	23	30	37	44	54	63	64	73	83	90	97	104	114	123
6	15	21	28	39	46	52	61	66	75	81	88	99	106	112	121
5	12	22	31	36	45	55	62	65	72	82	91	96	105	115	122
7	14	20	29	38	47	53	60	67	74	80	89	98	107	113	120

表20

0	9	19	26	33	40	50	59	68	77	87	94	101	108	118	127
2	11	16	25	35	42	49	56	70	79	84	93	103	110	117	124
1	8	18	27	32	41	51	58	69	76	86	95	100	109	119	126
3	10	17	24	34	43	48	57	71	78	85	92	102	111	116	125
4	13	23	30	37	44	54	63	64	73	83	90	97	104	114	123
6	15	20	29	39	46	53	60	66	75	80	89	99	106	113	120
5	12	22	31	36	45	55	62	65	72	82	91	96	105	115	122
7	14	21	28	38	47	52	61	67	74	81	88	98	107	112	121

表 21

0	4	18	22	33	37	51	55	73	77	91	95	104	108	122	126
1	5	19	23	32	36	50	54	72	76	90	94	105	109	123	127
2	6	16	20	35	39	49	53	75	79	89	93	106	110	120	124
3	7	17	21	34	38	48	52	74	78	88	92	107	111	121	125
9	13	27	31	40	44	58	62	64	68	82	86	97	101	115	119
8	12	26	30	41	45	59	63	65	69	83	87	96	100	114	118
11	15	25	29	42	46	56	60	66	70	80	84	99	103	113	117
10	14	24	28	43	47	57	61	67	71	81	85	98	102	112	116

表 22

0	4	18	22	33	37	51	55	73	77	91	95	104	108	122	126
1	5	19	23	32	36	50	54	72	76	90	94	105	109	123	127
2	6	17	21	35	39	48	52	75	79	88	92	106	110	121	125
3	7	16	20	34	38	49	53	74	78	89	93	107	111	120	124
9	13	27	31	40	44	58	62	64	68	82	86	97	101	115	119
8	12	26	30	41	45	59	63	65	69	83	87	96	100	114	118
11	15	24	28	42	46	57	61	66	70	81	85	99	103	112	116
10	14	25	29	43	47	56	60	67	71	80	84	98	102	113	117

表 23

0	4	19	23	33	37	50	54	73	77	90	94	104	108	123	127
1	5	18	22	32	36	51	55	72	76	91	95	105	109	122	126
2	6	17	21	35	39	48	52	75	79	88	92	106	110	121	125
3	7	16	20	34	38	49	53	74	78	89	93	107	111	120	124
9	13	26	30	40	44	59	63	64	68	83	87	97	101	114	118
8	12	27	31	41	45	58	62	65	69	82	86	96	100	115	119
11	15	24	28	42	46	57	61	66	70	81	85	99	103	112	116
10	14	25	29	43	47	56	60	67	71	80	84	98	102	113	117

表 24

0	4	19	23	33	37	50	54	73	77	90	94	104	108	123	127
1	5	18	22	32	36	51	55	72	76	91	95	105	109	122	126
2	6	16	20	35	39	49	53	75	79	89	93	106	110	120	124
3	7	17	21	34	38	48	52	74	78	88	92	107	111	121	125
9	13	26	30	40	44	59	63	64	68	83	87	97	101	114	118
8	12	27	31	41	45	58	62	65	69	82	86	96	100	115	119
11	15	25	29	42	46	56	60	66	70	80	84	99	103	113	117
10	14	24	28	43	47	57	61	67	71	81	85	98	102	112	116

经过分组交织生成 32 条目标数据流，对 32 条目标数据流中每个目标符号子集分别进行内码编码，其中内码编码信息位长度为 160 比特。具体地，内码编码器分别将目标数据流中每个目标符号子集总共 160 比特编码添加冗余，得到 32 条编码数据流。在一种可能的实施方式中，采用 Hamming(170,160)进行内码编码，分别将每条目标数据流中连续的 16 个 RS 符号总共 160 比特添加 10 比特冗余得到 170 比特的码字。在另一种可能的实施方式中，采用

BCH(176,160) 进行内码编码，分别将每条目标数据流中连续的 16 个 RS 符号总共 160 比特添加 16 比特冗余得到 176 比特的码字。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。

经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括调制映射（mapping），信道交织（channel interleaving），偏振划分（polarization distribution），或 DSP 成帧处理（framing）等。比如内码编码数据流可通过交织，以提升系统抗突发错误的能力。

采用本实施例的数据交织编码方案，当内码采用 Hamming(170,160)时，其 KP4 RS(544,514) + Hamming(170,160)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 4.3E-3，性能逼近该级联 FEC 方案的最优性能。当内码采用 BCH(176,160)时，其 KP4 RS(544,514) + BCH(176,160)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 8.3E-3，性能逼近该级联 FEC 方案的最优性能。

图 42 为本申请实施例提供的交织处理的另一种流程示意图。

4201、对 n 条通道数据流进行第一分组交织得到 T 条第一数据流。

本实施例中，经过第一分组交织后的每条第一数据流中连续的 C 个符号至少来自 E 个不同的码字，T=n/K1，C 为 a 的倍数，E≥K2*a。下面介绍第一分组交织的一种具体实现方式。

图 43 为本申请实施例中进行第一分组交织的一种实现方式示意图。如图 43 所示，参与第一分组交织的 n 条通道数据流包括第三符号矩阵，第三符号矩阵包括 n 行 A 列个符号，第三符号矩阵中不同行的 A 个符号分别来自不同的通道数据流，A 为 a 的倍数。经过第一分组交织后的 T 条第一数据流包括第四符号矩阵，所述第四符号矩阵包括 T 行 C 列个符号，第四符号矩阵中不同行的 C 个符号分别来自不同的第一数据流，T 为 n 的约数，n*A=T*C。具体地，第三符号矩阵的一列中每连续 T 个符号为符号子矩阵，第四符号矩阵中每一列的 T 个符号分别与第三符号矩阵中每个符号子矩阵一一对应。

在一种可能的实施方式中，第三符号矩阵中的符号子矩阵按第一顺序排列，第三符号矩阵中每列的第 1 行到第 n 行包括按第一顺序排列的第 1 个符号子矩阵到第 n/T 个符号子矩阵，第三符号矩阵的相邻两列中前一列的第 n/T 个符号子矩阵到后一列的第 1 个符号子矩阵为按第一顺序排列的连续 2 个符号子矩阵，第四符号矩阵中第 1 列的 T 个符号来自第三符号矩阵中按第一顺序排列的第一个符号子矩阵，依次类推，直到第四符号矩阵中第 C 列的 T 个符号来自第三符号矩阵中按第一顺序排列的最后一个符号子矩阵。也就是说，第三符号矩阵按照每一列从上到下依次将从左到右每一列的符号子矩阵映射到第四符号矩阵中，即先从上到下后从左到右的顺序映射。

在另一种可能的实施方式中，第三符号矩阵中的符号子矩阵按第二顺序排列，第三符号矩阵中每 T 行的第 1 列到第 A 行包括按第二顺序排列的第 1 个符号子矩阵到第 A 个符号子矩阵，第三符号矩阵的相邻两个连续 T 行中前 T 行的第 A 个符号子矩阵到后 T 行的第 1 个符号子矩阵为按第二顺序排列的连续 2 个符号子矩阵，第四符号矩阵中第 1 列的 T 个符号来自第三符号矩阵中按第二顺序排列的第一个符号子矩阵，依次类推，直到第四符号矩阵中第 C 列的 T 个符号来自第三符号矩阵中按第二顺序排列的最后一个符号子矩阵。也就是说，第三符

号矩阵按照同一行从左到右依次将从上到下共 n/T 行个符号子矩阵映射到第四符号矩阵中，即先从左到右后从上到下的顺序映射。

需要说明的是，在实际应用中，上述的第三符号矩阵和第四符号矩阵也可能不是以矩阵形式呈现的。例如，第三符号矩阵呈现为第三符号集合，第三符号集合包括 n*A 个符号子集，分别对应第三符号矩阵中的 n 行 A 列个符号。第四符号矩阵呈现为第四符号集合，第四符号集合包括 T*C 个符号子集，分别对应第四符号矩阵中的 T 行 C 列个符号。

4202、对 T 条第一数据流进行卷积交织得到 T 条第二数据流。

本实施例中，每条第二数据流中连续的 H 个符号来自至少 F 个不同码字，F≥E，每条第二数据流中连续的 H 个符号最多 K1/K2 个符号来自相同码字。

需要说明的是，本实施例中卷积交织的实现方式与上述图 10 所示实施例中介绍的卷积交织方式类似，此处不再赘述。一个区别在于，上述图 10 所示实施例中每个存储单元用于存储 d 个符号，本实施例中每个存储单元用于存储 C 个符号。在一种可能的实施方式中，若采用上述图 12（a）所示的卷积交织器，则卷积交织器的参数应满足$C(p*Q+1)\geq a*N*K1/(n/k2)$，以使得卷积交织器输出的第一数据流中连续的 H=p*C 个符号至少来自 F 个不同的外码码字，其中，N 为外码编码的码字长度。在另一种可能的实施方式中，若采用上述图 12（b）所示的卷积交织器，则卷积交织器的参数应满足$C(p*Q-1)\geq a*N*K1/(n/k2)$，以使得卷积交织器输出的第一数据流中连续的 H=p*C 个符号至少来自 F 个不同的外码码字，其中，N 为外码编码的码字长度。

4203、对 T 条第二数据流中每条第二数据流进行第二分组交织得到 S 条目标数据流，以得到共 m 条目标数据流。

本实施例中，m=T*S，$S\geq k1/K2$。下面介绍第二分组交织的一种具体实现方式。图 44（a）为本申请实施例中进行第二分组交织的一种实现方式示意图。如图 44（a）所示，每个第二分组交织器对输入的第二数据流进行分组交织得到 S 条目标数据流，从而共产生 m=T*S 条目标数据流。

图 44（b）为本申请实施例中进行第二分组交织的一种具体实现方式示意图。如图 44（b）所示，每条第二数据流包括 R 个符号集合，每个符号集合包括 p 个符号子集，每个符号子集包括 C 个符号，p 个符号子集分别由 p 条延迟线输出，每个符号集合中的符号来自至少 F 个不同码字，每条目标数据流包括 F 个符号，R*p*C=S*F，R 为大于或等于的整数。也就是说，由卷积交织器 p 条延迟线轮询 R 次输出的 R 个符号集合进行第二分组交织得到 S 条目标数据流。应理解，一次分组交织操作只是得到目标数据流中 F 个连续的符号，连续不断的进行分组交织才是得到目标数据流，目标数据流中连续的 F 个符号表示为目标符号子集。目标数据流中每连续的 F 个符号来自至少$\lfloor F/K2*a)\rfloor$个不同的符号子集，且$\lceil F/(K2*a)\rceil$个不同的符号子集中每个符号子集最多取$K2*a$个符号，$\lceil F/(K2*a)\rceil$表示对 $F/(K2*a)$的商进行向上取整的整数。还应理解，每条目标数据流中每连续的 F 个符号最多有$\lfloor F/R\rfloor$个符号来自相同的符号集合，$\lfloor F/R\rfloor$表示对 F/R的商进行向上取整的整数。

在一种可能的实施方式中，目标数据流中每连续的 F 个符号包括来自第一符号子集中的第一组符号和来自第二符号子集的第二组符号，第一符号子集和第二符号子集属于同一个符

号集合。第一符号子集和第二符号子集分别由两条相邻的延迟线输出，第一符号子集中的符号和第二符号子集中的符号分别按次序排列，第一组符号在第一符号子集中的次序与第二组符号在第二符号子集中的次序不同。也就是说，第一组符号和第二组符号分别在各自符号子集中的位置不同。

在另一种可能的实施方式中，目标数据流每连续的F个符号包括来自第三符号子集中的第三组符号和来自第四符号子集的第四组符号，第三符号子集和第四符号子集属于不同的符号集合，第三符号子集和第四符号子集由同一条延迟线输出，第三符号子集中的符号和第四符号子集中的符号分别按次序排列，第三组符号在第三符号子集中的次序与第四组符号在第四符号子集中的次序不同。也就是说，第三组符号和第四组符号分别在各自符号子集中的位置不同。

本发明针对采用级联FEC的传输方案，设计一种包含第一分组交织、卷积交织、第二分组交织和编码的数据交织编码方法，可以对所有接入业务采用同样的交织方案，使得整体级联FEC方案性能较好且时延较低。使得该级联FEC传输方案能够应用于较多的传输场景，尤其适用于对需要较低传输时延的传输场景，如低时延数据中心互联场景。

下面结合一些具体的实施例对上述图42所描述的交织处理方法的流程进行进一步介绍。应理解，下述每个实施例中的内码并行度指的是目标数据流的数量，例如，内码并行度16表示目标数据流的数量是16。还应理解，下述各实施例中的目标符号子集均表示目标数据流中连续的F个符号。

实施例1：内码编码信息位长度为120比特，内码并行度16。

发端处理模块根据接入业务的类型，利用各业务PCS通道或者FEC通道已知的对齐标识（Alignment marker）进行通道数据流的标识锁定（alignment lock）。这里32个通道的已知对齐标识各不相同，且与接入的业务相关。发端处理模块接着对每个业务内的多个PCS通道或者FEC通道进行纠偏处理，纠偏处理后同一个业务内的多个PCS/FEC通道之间都是AM对齐，而业务之间的通道数据流仅需要满足RS符号对齐，即AM之间相差整数个RS符号。比如当接入业务为2*400GbE时，纠偏处理后32条通道数据流如图37所示，每个400GbE业务内16条PCS通道之间都为AM对齐，而2个业务之间的AM相差整数个RS符号。然后根据对齐标识对n=32条通道的数据进行通道重排序（lane reorder）处理，使得n=32条通道的数据能够按照指定的顺序排列。如当接入业务为1*800GbE，32条通道数据流如图5所示，如当接入业务为2*400GbE，32条通道数据流如图6所示，如当接入业务为4*200GbE，32条通道数据流如图7所示；如当接入业务为8*100GbE且为“100G RS-FEC-Int”模式，32条通道数据流如图8所示；如当接入业务为8*100GbE且为“100G RS-FEC”模式时，32条通道数据流如图9所示。

经通道重排序的n=32条通道数据流送入所设计的第一分组交织模块、卷积交织模块和第二分组交织模块进行相应处理后送入内码编码器进行内码编码；经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括调制映射（mapping），信道交织（channel interleaving），偏振划分（Polarization distribution），或DSP成帧处理

（Framing）等。

在本实施例中，根据通道数据流中RS分布规律，将n个通道数据流划分为G个通道数据流子集，任意两个通道数据流子集来自不同的RS码字，因此根据接入的业务的不同，其通道数据流包含的通道数据流子集数目不同，结合如5到图9所示PCS/FEC通道数据流格式，如当接入业务为1*800GbE或者2*400GbE，32条通道数据流包含G=2个通道数据流子集，即通道数据流0到15为一个通道流子集，通道数据流16到31为一个通道流子集；如当接入业务为4*200GbE，32条通道数据流包含G=4个通道数据流子集；如当接入业务为1*800GbE，32条通道数据流包含G=8个通道数据流子集，则32条通道数据流最多包含$G_{max} = n/4 = 8$个通道数据流子集，最少包含$G_{min} = n/16 = 2$个通道数据流子集，要使得第一分组交织对接入业务不敏感且整体交织时延最短，则第一交织模块对应参数K1=$G_{max} = 8$，T=n/K1=4。

图45（a）为第一分组交织的一种实现方式示意图。如图45（a）所示，从每条通道数据流取连续A=2个符号，构成32行2列的第三符号矩阵，进行分组交织得到T=4行C=16列的第四符号矩阵，第四符号矩阵第t行的C=16个符号为第一数据流t上连续的16个符号，其中$0 \le t < 3$。第四符号矩阵第i行第j列的符号来自第三符号矩阵的第x行第y列，其中满足$x = j * 4\%32 + i, y = \lfloor j * 4/32 \rfloor, (0 \le i < 4, 0 \le j < 16)$，其中$\lfloor j * 4/32 \rfloor$表示为对$j * 4/32$向下取整。定义第四符号矩阵一行为一个符号子集，并结合图5到图9所示PCS/FEC通道数据流格式，可以得到每个符号子集的符号来自于K1=8个不同的通道数据流，8个通道数据流可以表示为通道数据流j_0,通道数据流j_1,通道数据流j_2,通道数据流j_3,通道数据流j_4,通道数据流j_5,通道数据流j_6,通道数据流j_7,且$\lfloor j_x/4 \rfloor = x, x \in [0,7]$,$\lfloor j_x/4 \rfloor$表示对$j_x/4$向下取整，每个符号子集至少来自于$K2 * a = 4$个不同RS码字，最多来自于$K1 * a$=16个不同RS码字，其中K2=$G_{min}$。

经过第一分组交织模块得到的4条第一数据流送入到卷积交织模块，得到4条第二数据流。卷积交织模块包含4个卷积交织器，分别表示为卷积交织0、卷积交织1、卷积交织2、卷积交织3，其采用相同的交织结构，每个交织交织对一条第一数据流进行交织处理得到一条第二数据流。

图45（b）为卷积交织的一种实现方式示意图。如图45（b）所示，其包含p=3条延迟线（delay line）。这p=3条延迟线分别包括2Q个存储单元、Q个存储单元、0个存储单元，每个存储单元为d=C=16个符号，即用于存储第一数据流中一个符号子集。也就是，延迟线0的延迟取值为32Q个符号，延迟线1的延迟取值为16Q个符号，延迟线1的延迟取值为0个符号即无延迟。

如图45（b）所示的$S_r()$表示第一数据流r（$0 \le r \le T-1$）中一个符号子集，包含C=16个符号。比如$S_r(3t)$表示第一数据流r当前输入给延迟线0的一个符号子集，且$S_r(3t-6Q)$为延迟线0输出的一个符号子集；$S_r(3t+1)$表示第一数据流r当前输入给延迟线1的一个符号子集，且$S_r(3t-3Q+1)$为延迟线1输出的一个符号子集；$S_r(3t+2)$表示第一数据流r当前输入给延迟线2的一个符号子集，且$S_r(3t+2)$为延迟线2输出的一个符号子集；$S_r(3t+3)$表示第一数据流r当前输入给延迟线0的一个符号子集，且$S_r(3t-6Q+4)$为延迟线0输出的一个符号子集；依次类推。结合图5和6，可以得出当接入业务为1*800GbE或者2*400GbE的时候，

$C(p*Q+1) \geq a*N*K1/b = 544$，即$Q \geq 11$时，卷积交织连续输出的$S_r(3t-6Q)$，$S_r(3t-3Q+1)$，$S_r(3t+2)$三个符号子集中包含 12 个不同的 RS 码字。结合图 7 和图 9，可以看出当接入业务为 4*200GbE 或者 8*100GbE 且为“100G RS-FEC”模式的时候，$C(p*2*Q+2) \geq a*N*K1/b = 1088$，即$Q \geq 11$时，卷积交织连续输出的$S_r(3t-6Q)$，$S_r(3t-3Q+1)$，$S_r(3t+2)$三个符号子集中包含 16 个不同的 RS 码字。结合图 8，可以看出当接入业务为 8*100GbE 且为“100G RS-FEC-int”模式的时候，$Q \geq 0$时，卷积交织连续输出的$S_r(3t-6Q)$，$S_r(3t-3Q+1)$，$S_r(3t+2)$三个符号子集中包含 16 个不同的 RS 码字；所以当$Q > 11$得时候可以保证对所有业务接入保证了卷积交织连续输出的$S_r(3t-6Q)$，$S_r(3t-3Q+1)$，$S_r(3t+2)$三个符号子集中至少包含 12 个不同的 RS 码字。

图 45（c）为卷积交织的另一种实现方式示意图。如图 45（c）所示，在一种可能的实施方式中，选取$Q = 11$，其对应的交织时延约为 22*16*3/2=528 个 RS 符号。

采用如图 45（c）所示的卷积交织器分别对 4 条第四数据数据流进行卷积交织处理得到 4 条第二数据流。定义第二数据流中从卷积交织器的延迟线 0，延迟线 1、延迟线 2 分别输出的$S_r(3t-6Q)$，$S_r(3t-3Q+1)$，$S_r(3t+2)$三个符号子集为符号集合，共总共包含 48 个 RS 符号，每个符号子集中的 16 个符号依次表示为符号 0 到符号 15。结合图 5 到图 9 中给出各种业务的 PCS/FEC 通道上 RS 码字的分布以及第一分组交织和卷积交织不难理解每个符号子集至少来自于 4 个不同的 RS 码字，且同一符号子集的符号 0 到符号 3 对应的 RS 码字与符号 4 到符号 7 对应的 RS 码字不同，同一符号子集的符号 8 到符号 11 对应的 RS 码字和符号 12 到符号 15 对应的 RS 码字不同，同一符号子集的符号 0 到符号 7 对应的 RS 码字与符号 8 到符号 15 对应的 RS 码字可能相同；进一步符号集合至少来自于 12 个不同 RS 码字，且符号集合中最多 K1/K2=4 个符号属于同一个 RS 码字；进一步任意 2 个符号子集的符号 0 到符号 7 各自对应的 RS 码字分布可能一致，任意 2 个符号子集的符号 8 到符号 15 各自对应的 RS 码字分布可能一致。

采用 4 个第二分组交织器分别对 4 条第二数据流进行交织得到 m=S*T=16 条目标数据流。图 45（d）为第二分组交织的一种实现方式的示意图，如图 45（d）所示，对于第二分组交织 i（0≤i<4）从第二数据流 i 中获取一个符号集合，从每个符号子集各取 4 个 RS 符号共 12 个 RS 符号映射为一个目标符号子集，总共得到 S=4 个目标符号子集，每个目标符号子集中的 12 个符号来自于 12 个不同的 RS 码字，4 个目标符号子集分别为分组交织器 i 输出的目标数据流 i*S 到目标数据流 i*S+3 上的连续 12 个 RS 符号。为表述方便定义$R(x,y)$为符号子集 x 的符号 y（$x \in [0,2], y \in [0,15]$），对于来自于任意 2 个符号子集中的各自任一个符号$R(x_1,y_1)$和$R(x_2,y_2)$，$x_1 \neq x_2$，如果$y_1 \neq y_2$，表示此 2 个符号位于符号子集的不同位置。为使得目标符号子集的 12 个 RS 符号来自于 12 个不同 RS 码字，则此 12 个 RS 符号对应于符号集合中的

$R(0,i_0), R(0,i_1), R(0,i_2), R(0,i_3), R(1,i_4), R(1,i_5), R(1,i_6), R(1,i_7)$，$R(2,i_8), R(2,i_9), R(2,i_{10})$，$R(2,i_{11})$；且满足$i_0, i_4, i_8$互不相同，$i_1, i_5, i_9$互不相同，$i_2, i_6, i_{10}$互不相同，$i_3, i_7, i_{11}$互不相同，$i_0, i_2\%8, i_4, i_6\%8$互不相同，$i_4, i_6\%8, i_8, i_{12}\%8$互不相同，$i_1, i_3\%8, i_5, i_7\%8$互不相同，$i_5, i_7\%8, i_9, i_{11}\%8$互不相同，$i_0, i_4, i_8 \in [0,3], i_1, i_5, i_9 \in [4,7], i_2, i_6, i_{10} \in [8,11], i_3, i_7, i_{11} \in [12,15]$；此等效为目标符号子集中的 12 个 RS 符号来自 3 个符号子集不同位置的符号，且每 4 个 RS 符

号来自同一个符号子集；进一步目标符号子集中来自同一个符号子集的4个符号分别为符号子集的符号0到符号3中的任一个，符号子集的符号4到符号7中的任一个，符号子集的符号8到符号11中的任一个，符号子集的符号12到符号15中的任一个；进一步目标符号子集中来自于由延迟取值差值为Q*C=176两个延迟线输出的符号子集的8个RS符号位于对应符号子集不同位置；进一步目标符号子集中最多有2个RS符号来自于同一个通道数据流，其通过第一分组交织映射到2个不同符号子集，且对应的2个符号子集由卷积交织器2个不同的延迟线输出，且对应2个延迟线的延迟差值大于等于2*Q*C=352个RS符号。

由此规则符号集合到目标符号子集的具体映射有很多种，表1-4给出了几种具体的映射关系，表1-4中每个表第y行第z列的数字x表示目标数据流y的中的第z个RS符号来自于符号子集$\lfloor x/16 \rfloor$的第x%16个RS符号，其中$0 \le y < 4, 0 \le z < 12, 0 \le x < 48$。需要说明的是将表中进行任意行之间交换，或者任意列之间交换，或者任意行交换后在进行任意列交换，或者先任意列交换后在进行任意行交换都是一个合法的映射。

根据每个符号子集与通道通道数据流的映射关系，以及符号集合中每个符号子集的时延关系，不难得出目标符号子集与通道数据流的关系为：目标符号子集的数据来源于K1=8个不同通道数据流，8个通道数据流可以表示为通道数据流j_0,通道数据流j_1,通道数据流j_2,通道数据流j_3,通道数据流j_4,通道数据流j_5,通道数据流j_6,通道数据流j_7,且$\lfloor j_x/4 \rfloor = x, x \in [0,7]$,$\lfloor j_x/4 \rfloor$表示对$j_x/4$向下取整；进一步从其中4个通道数据流取1个RS符号，从剩下4个通道数据流取2个RS符号；进一步如果从某一个通道数据流取2个RS符号，这2个RS符号在对应通道数据流间隔的距离大于a*N*K2/n，即大于等于2*544*2/32=68个RS符号；进一步最多2个RS符号来自2个不同通道数据流上对齐的2个RS符号。

经过上述分组交织得到的16个目标数据流分别送入到内码编码器，每个目标符号子集作为一个内码编码的信息数据，通过内码编码器产生冗余数据，得到16条编码数据流。在一种可能的实施方式中，采用Hamming(128,120)进行内码编码，分别将目标数据流中的每个目标符号子集的12个RS符号总共120比特添加8比特冗余得到128比特的码字。在另一种可能的实施方式中，采用BCH(136,120) 进行内码编码，分别将目标数据流中的每个目标符号子集的12个RS符号总共120比特添加16比特冗余得到136比特的码字。

经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括调制映射（mapping），信道交织（channel interleaving），偏振划分（polarization distribution），或DSP成帧处理（Framing）等。比如内码编码数据流可通过交织，以提升系统抗突发错误的能力。

采用本实施例的数据交织编码方案，其KP4 RS(544,514) + Hamming(128,120)级联码在AWGN下，为了达到纠后误比特率BER为1E-15所对应的纠前BER约为4.5E-3，性能逼近该级联FEC方案的最优性能。

实施例2：内码编码信息位长度为160比特，内码并行度16。

在本实施例中，采用实施例1中的第一分组交织方案对n个通道数据流进行第一分组交织处理，得到T=4条第一数据流，然后4个卷积交织器分别对4条第一数据流进行卷积交织

得到 4 条第二数据流。其中卷积交织 0、卷积交织 1、卷积交织 2、卷积交织 3 采用相同的交织结构。

图 46（a）为卷积交织的另一种实现方式示意图。如图 46（a）所示，其包含 p=4 条延迟线（delay line）。这 4 条延迟线分别包括 3Q 个存储单元、2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元为 d=C=16 个符号，即用于存储第一数据流中一个符号子集。也就是，延迟线 0 的延迟取值为 48Q 个符号，延迟线 1 的延迟取值为 32Q 个符号，延迟线 2 的延迟取值为 16Q 个符号，延迟线 3 的延迟取值为 0 个符号即无延迟。

如图 46(a)所示的 $S_r()$ 表示第一数据流$r(0 \le r \le T-1)$中一个符号子集。比如$S_r(4t)$表示第一数据流 r 当前输入给延迟线 0 的一个符号子集，且$S_r(4t-12Q)$为延迟线 0 输出的一个符号子集；$S_r(4t+1)$表示第一数据流 r 当前输入给延迟线 1 的一个符号子集，且$S_r(4t-8Q+1)$为延迟线 1 输出的一个符号子集；$S_r(4t+2)$表示第一数据流 r 当前输入给延迟线 2 的一个符号子集，且$S_r(4t-4Q+2)$为延迟线 2 输出的一个符号子集；$S_r(4t+3)$表示第一数据流 r 当前输入给延迟线 3 的一个符号子集，且$S_r(4t+3)$为延迟线 3 输出的一个符号子集；$S_r(4t+4)$表示第一数据流 r 当前输入给延迟线 0 的一个符号子集，且$S_r(4t-12Q+4)$为延迟线 0 输出的一个符号子集；依次类推。结合图 5 和图 6，可以得出当接入业务为 1*800GbE 或者 2*400GbE 的时候，$C(p*Q+1) \ge 544$即$Q \ge 9$时，卷积交织连续输出的$S_r(4t-12Q)$，$S_r(4t-8Q+1)$，$S_r(4t-4Q+2)$,$S_r(4t+3)$四个符号子集中包含 16 个不同的 RS 码字；结合图 7 和图 9，可以看出当接入业务为 4*200GbE 或者 8*100GbE 且为“100G RS-FEC”模式的时候，$C(p*2*Q+2) \ge 1088$即$Q \ge 9$时，卷积交织连续输出的$S_r(4t-12Q)$，$S_r(4t-8Q+1)$，$S_r(4t-4Q+2)$,$S_r(4t+3)$四个符号子集中包含 16 个不同的 RS 码字；结合图 8，可以看出当接入业务为 8*100GbE 且为“100G RS-FEC-int”模式的时候，$Q \ge 0$时，卷积交织连续输出的$S_r(4t-12Q)$，$S_r(4t-8Q+1)$，$S_r(4t-4Q+2)$,$S_r(4t+3)$四个符号子集中包含 16 个不同的 RS 码字；所以当$Q \ge 9$的时候可以保证对所有业务接入卷积交织连续输出的$S_r(4t-12Q)$，$S_r(4t-8Q+1)$，$S_r(4t-4Q+2)$,$S_r(4t+3)$四个符号子集中包含 16 个不同的 RS 码字。

图 46（b）为卷积交织的另一种实现方式示意图。如图 46（b）所示，在一种可能的实施方式中，选取$Q=9$，其对应的交织时延约为 27*16*3/2=648 个 RS 符号。

采用 4 个如图 46（b）所示的卷积交织器分别对 4 条第四数据数据流进行卷积交织处理得到 4 条第二数据流。定义第二数据流中从卷积交织器的延迟线 0，延迟线 1、延迟线 2 和延迟线 3 分别输出的$S_r(4t-12Q)$，$S_r(4t-8Q+1)$，$S_r(4t-4Q+2)$，$S_r(4t+3)$四个符号子集为符号集合，其总共包含 64 个 RS 符号，每个符号子集中的 16 个符号依次表示为符号 0 到符号 15。结合图 5 到图 9 中给出各种业务的 PCS/FEC 通道上 RS 码字的分布以及第一分组交织和卷积交织不难理解每个符号子集至少来自于 4 个不同的 RS 码字，且同一符号子集的符号 0 到符号 3 对应的 RS 码字与符号 4 到符号 7 对应的 RS 码字不同，同一符号子集的符号 8 到符号 11 对应的 RS 码字和符号 12 到符号 15 对应的 RS 码字不同，同一符号子集的符号 0 到符号 7 对应的 RS 码字与符号 8 到符号 15 对应的 RS 码字可能相同；进一步符号集合来自于 16 个不同 RS 码字，且符号集合中最多 K1/K2=4 个符号属于同一个 RS 码字；进一步任意 2 个符号子集的符号

0 到符号 7 各自对应的 RS 码字分布可能一致，任意 2 个符号子集的符号 8 到符号 15 各自对应的 RS 码字分布可能一致。

采用 4 个第二分组交织器分别对 4 条第二数据流进行交织得到 m=S*T=16 条目标数据流。图 46（c）为第二分组交织的另一种实现方式示意图。如图 46（c）所示，对于第二分组交织 i（0≤i<4）从第二数据流 i 中获取一个符号集合，从每个符号子集各取 4 个 RS 符号共 16 个 RS 符号映射为一个目标符号子集，总共得到 S=4 个目标符号子集，每个目标符号子集中的 16 个符号来自于 16 个不同的 RS 码字，4 个目标符号子集分别为分组交织器 i 输出的目标数据流 i*S 到目标数据流 i*S+3 上的连续 16 个 RS 符号。为表述方便定义$R(x,y)$为符号子集 x 的符号 y（$x \in [0,3], y \in [0,15]$）。为使得目标符号子集中的 16 个 RS 符号来自于 16 个不同 RS 码字，则此 16 个 RS 符号对应于符号集中的

$R(0,i_0), R(0,i_1), R(0,i_2), R(0,i_3), R(1,i_4), R(1,i_5) R(1,i_6), R(1,i_7)$，$R(2,i_8), R(2,i_9), R(2,i_{10}),$

$R(2,i_{11}), R(3,i_{12}), R(3,i_{13}), R(3,i_{14}), R(3,i_{15})$；且满足$i_0, i_4, i_8, i_{12}$互不相同，$i_1, i_5, i_9, i_{13}$互不相同，$i_2, i_6, i_{10}, i_{14}$互不相同，$i_3, i_7, i_{11}, i_{15}$互不相同，$i_0, i_2\%8, i_4, i_6\%8$互不相同，$i_4, i_6\%8, i_8, i_{12}\%8$互不相同，$i_8, i_{12}\%8, i_{14}, i_{16}\%8$互不相同，$i_1, i_3\%8, i_5, i_7\%8$互不相同，$i_5, i_7\%8, i_9, i_{11}\%8$互不相同，$i_9, i_{11}\%8, i_{13}, i_{15}\%8$互不相同，$i_0, i_4, i_8, i_{12} \in [0,3], i_1, i_5, i_9, i_{13} \in [4,7], i_2, i_6, i_{10}, i_{14} \in [8,11], i_3, i_7, i_{11}, i_{15} \in [12,15]$；此等效为目标符号子集中的 16 个 RS 符号来自 4 个符号子集不同位置的符号，且每 4 个 RS 符号来自同一个符号子集；进一步目标符号子集中来自同一个符号子集的 4 个符号分别为符号子集的符号 0 到符号 3 中的任一个，符号子集的符号 4 到符号 7 中的任一个，符号子集的符号 8 到符号 11 中的任一个，符号子集的符号 12 到符号 15 中的任一个；进一步目标符号子集中来自于由延迟取值差值为 Q*C=144 的两个延迟线输出的符号子集的 8 个 RS 符号位于对应符号子集不同位置；进一步目标符号子集中最多有 2 个 RS 符号来自于同一个通道数据流，其通过第一分组交织映射到 2 个不同符号子集，且对应的 2 个符号子集由卷积交织器 2 个不同的延迟线输出，且对应 2 个延迟线的延迟差值大于等于 2*Q*C=288 个 RS 符号。

由此规则符号集合到目标符号子集的具体映射有很多种，表 13-16 给出了几种具体的映射关系，每个表第 y 行第 z 列的数字 x 表示目标符号子集 y 的中的第 z 个 RS 符号来自于符号子集$\lfloor x/16 \rfloor$的第 x%16 个 RS 符号，其中$0 \le y < 4, 0 \le z < 16, 0 \le x < 64$。需要说明的是将表中进行任意行之间交换，或者任意列之间交换，或者任意行交换后在进行任意列交换，或者先任意列交换后在进行任意行交换都是一个合法的映射。

根据每个符号子集与通道通道数据流的映射关系，以及符号集合中每个符号子集的时延关系，不难得出目标符号子集的数据与通道数据流的关系为：目标符号子集的数据来源于 K1=8 个不同通道数据流，8 个通道数据流可以表示为通道数据流j_0,通道数据流j_1,通道数据流j_2,通道数据流j_3,通道数据流j_4,通道数据流j_5,通道数据流j_6,通道数据流j_7,且$\lfloor j_x/4 \rfloor = x, x \in [0,7]$,$\lfloor j_x/4$ 表示对 $jx/4$ 向下取整；进一步从每个通道数据流取 2 个 RS 符号；此 2 个 RS 符号在对应通道数据流间隔的距离大于等于 a*N*K2/n，即大于等于 2*544*2/32=68 个 RS 符号；进一步最多 2 个 RS 符号来自 2 个不同通道数据流上对齐的 2 个 RS 符号。

经过上述分组交织得到的 16 个目标数据流分别送入到内码编码器，每个目标符号子集作

为一个内码编码的信息数据，通过内码编码器产生冗余数据，得到16条编码数据流。在一种可能的实施方式中，采用Hamming(170,160)进行内码编码，分别将每个目标符号子集的16个RS符号总共160比特添加10比特冗余得到170比特的码字。在另一种可能的实施方式中，采用BCH(176,160) 进行内码编码，分别将每个目标符号子集的16个RS符号总共160比特添加16比特冗余得到176比特的码字。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。

采用本实施例的数据交织编码方案，当内码采用Hamming(170,160)时，其KP4 RS(544,514) + Hamming(170,160)级联码在AWGN下，为了达到纠后误比特率BER为1E-15所对应的纠前BER约为4.3E-3，性能逼近该级联FEC方案的最优性能。当内码采用BCH(176,160)时，其KP4 RS(544,514) + BCH(176,160)级联码在AWGN下，为了达到纠后误比特率BER为1E-15所对应的纠前BER约为8.3E-3，性能逼近该级联FEC方案的最优性能。

实施例3：内码编码信息位长度为120比特，内码并行度32。

在本实施例中，采用实施例1中的第一分组交织方案对n个通道数据流进行第一分组交织处理，得到T=4条第一数据流，然后采用如图45（c）所示的卷积交织器分别对4条第一数据数据流进行卷积交织处理得到4条第二数据流。定义第二数据流中从卷积交织器的延迟线0，延迟线1、延迟线2分别输出的$S_r(3t-6Q)$，$S_r(3t-3Q+1)$，$S_r(3t+2)$三个符号子集为符号集合。从每个第二数据流中取连续Q=2个符号集合，标记为符号集合0和符号集合1，其总共包含6个符号子集，分别表示为符号子集0、符号子集1、符号子集2、符号子集3、符号子集4和符号子集5。第二分组交织i（$0\leqslant i<4$）将第二数据流i中2个连续符号集合进行分组交织得到8个目标符号子集，每个目标符号子集包含12个RS符号，8个目标符号子集分别为分组交织器i输出目标数据流i*S到目标数据流i*S+7上的连续12个RS符号，且目标符号子集中的每个RS符号来自不同的RS码字。结合图5到图9中给出各种业务的PCS/FEC通道上RS码字的分布以及第一分组交织和卷积交织，不难理解每个符号子集至少来自于4个不同的RS码字，且同一符号子集的符号0到符号3对应的RS码字与符号4到符号7对应的RS码字不同，同一符号子集的符号8到符号11对应的RS码字和符号12到符号15对应的RS码字不同，同一符号子集的符号0到符号7对应的RS码字与符号8到符号15对应的RS码字可能相同；进一步任意2个符号子集的符号0到符号7对应RS码字分布可能一致，任意2个符号子集的符号8到符号15对应RS码字分布可能一致；进一步每个符号集合至少来自于12个不同RS码字，且符号集合中最多4个符号属于同一个RS码字，符号集合0和符号集合1中RS码字分布一致。

图47为第二分组交织的另一种实现方式示意图。如图47所示，从每个符号子集各取2个RS符号共12个RS符号映射为一个目标符号子集，总共得到8个目标符号子集。为表述方便定义$R(x,y)$为符号子集x的符号y（$x\in[0,5],y\in[0,15]$）。为使得目标符号子集中的12个RS符号来自于12个不同RS码字，则此12个RS符号对应于2个符号集合中的

$R(0,i_0),R(0,i_1),R(1,i_2),R(1,i_3),R(2,i_4),R(2,i_5)$，$R(3,i_6),R(3,i_7),R(4,i_8),R(4,i_9),R(5,i_{10})$，$R(5,i_{11})$；且满足$i_0,i_1,i_2,i_3,i_4,i_5$，$i_6,i_7,i_8,i_9,i_{10},i_{11}$互不相同，$i_0\%8,i_1\%8,i_2\%8,i_3\%8$互

不相同，$i_2\%8, i_3\%8, i_4\%8, i_5\%8$互不相同，$i_6\%8, i_7\%8, i_8\%8, i_9\%8$互不相同，$i_8\%8, i_9\%8, i_{10}\%8, i_{11}\%8$互不相同，$i_0\%8, i_1\%8, i_6\%8, i_7\%8$互不相同，$i_2\%8, i_3\%8, i_8\%8, i_9\%8$互不相同，$i_4\%8, i_5\%8, i_{10}\%8, i_{11}\%8$互不相同，$i_0, i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10}, i_{11} \in [0,15]$；此等效为目标符号子集中的 12 个 RS 符号来自 6 个符号子集不同位置的符号，且每 2 个 RS 符号来自同一个符号子集；进一步目标符号子集中来自于由延迟取值差值为 Q*C=176 的两个延迟线输出的符号子集的 4 个 RS 符号位于对应符号子集不同位置；进一步目标符号子集中来自于卷积交织器同一延迟线输出的 2 个符号子集的 4 个 RS 符号位于对应符号子集不同位置；进一步目标符号子集中最多有 2 个 RS 符号来自于同一个通道数据流，其通过第一分组交织映射到 2 个不同符号子集，且对应的 2 个符号子集由卷积交织器 2 个不同的延迟线输出，且对应 2 个延迟线的延迟差值大于等于 2*Q*C=352 个 RS 符号。由此规则 2 个符号集合交织到 8 个目标符号子集的具体映射有很多种，表 5-12 给出了几种具体的映射关系，每个表第 y 行第 z 列的数字 x 表示目标符号子集 y 的中的第 z 个 RS 符号来自于符号子集$\lfloor x/16 \rfloor$的第 x%16 个 RS 符号，其中$0 \le y < 8, 0 \le z < 12, 0 \le x < 96$。需要说明的是将表中进行任意行之间交换，或者任意列之间交织，或者任意行交织后在进行任意列交换，或者先任意列交换后在进行任意行交换都是一个合法的映射。

根据每个符号子集与通道通道数据流的映射关系，以及符号集合中每个符号子集的时延关系，不难得出目标符号子集与通道数据流的关系为：目标符号子集的数据来源于 K1=8 个不同通道数据流，8 个通道数据流可以表示为通道数据流j_0,通道数据流j_1,通道数据流j_2,通道数据流j_3,通道数据流j_4,通道数据流j_5,通道数据流j_6,通道数据流j_7,且$\lfloor j_x/4 \rfloor = x, x \in [0,7]$,$\lfloor j_x/4 \rfloor$表示对$j_x/4$向下取整；进一步从其中 4 个通道数据流取 1 个 RS 符号，从剩下 4 个通道数据流取 2 个 RS 符号；进一步如果从某一个通道数据流取 2 个 RS 符号，这 2 个 RS 符号在对应通道数据流间隔的距离大于等于 2*544*2/32=68 个 RS 符号；进一步最多 2 个 RS 符号来自 2 个不同通道数据流上对齐的 2 个 RS 符号。

经过上述分组交织得到的 32 个目标符号子集分别送入到内码编码器，每个目标符号子集作为一个内码编码的信息数据，通过内码编码器产生冗余数据，得到 32 条编码数据流。在一种可能的实施方式中，采用 Hamming(128,120)进行内码编码，分别将目标符号子集的 12 个 RS 符号总共 120 比特添加 8 比特冗余得到 128 比特的码字。在另一种可能的实施方式中，采用 BCH(136,120) 进行内码编码，分别将目标符号子集的 12 个 RS 符号总共 120 比特添加 16 比特冗余得到 136 比特的码字。

经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括调制映射（mapping），信道交织（channel interleaving），偏振划分（polarization distribution），或 DSP 成帧处理（Framing）等。比如内码编码数据流可通过交织，以提升系统抗突发错误的能力。

实施例 4：内码编码信息位长度为 160 比特，内码并行度 32。

采用实施例 1 中的第一分组交织方案对 n 个通道数据流进行第一分组交织处理，得到 T=4 条第一数据流，采用如图 46（b）所示的卷积交织器分别对 4 条第四数据数据流进行卷积交

织处理得到 4 条第二数据流。定义第二数据流中从卷积交织器的延迟线 0，延迟线 1、延迟线 2 和延迟线 3 分别输出的$S_r(4t-12Q)$，$S_r(4t-8Q+1)$，$S_r(4t-4Q+2)$，$S_r(4t+3)$四个符号子集为符号集合，其总共包含 64 个 RS 符号。第二分组交织 i（$0\leqslant i<4$）将第二数据流 i 中 2 个连续符号集合进行分组交织得到 8 个目标符号子集，每个目标符号子集包含 16 个 RS 符号，8 个目标符号子集分别为分组交织器 i 输出目标数据流 i*S 到目标数据流 i*S+7 上的连续 16 个 RS 符号，且目标符号子集中的每个 RS 符号来自不同的 RS 码字。结合图 5 到图 9 中给出各种业务的 PCS/FEC 通道上 RS 码字的分布以及第一分组交织和卷积交织，不难理解每个符号子集至少来自于 4 个不同的 RS 码字，且同一符号子集的符号 0 到符号 3 对应的 RS 码字与符号 4 到符号 7 对应的 RS 码字不同，同一符号子集的符号 8 到符号 11 对应的 RS 码字和符号 12 到符号 15 对应的 RS 码字不同，同一符号子集的符号 0 到符号 7 对应的 RS 码字与符号 8 到符号 15 对应的 RS 码字可能相同；进一步任意 2 个符号子集的符号 0 到符号 7 对应 RS 码字分布可能一致，任意 2 个符号子集的符号 8 到符号 15 对应 RS 码字分布可能一致；进一步每个符号集合至少来自于 12 个不同 RS 码字，且符号集合中最多 4 个符号属于同一个 RS 码字，符号集合 0 和符号集合 1 中 RS 码字分布一致。

图 48 为第二分组交织的另一种实现方式示意图。如图 48 所示，从每个符号子集各取 2 个 RS 符号共 16 个 RS 符号映射为一个目标符号子集，总共得到 8 个目标符号子集。为表述方便定义$R(x,y)$为符号子集 x 的符号 y（$x\in[0,7],y\in[0,15]$）。为使得目标符号子集中的 16 个 RS 符号来自于 16 个不同 RS 码字，则此 16 个 RS 符号对应于 2 个符号集合中的 $R(0,i_0),R(0,i_1),R(1,i_2),R(1,i_3),R(2,i_4),R(2,i_5),R(3,i_6),R(3,i_7),R(4,i_8),R(4,i_9),R(5,i_{10})$，$R(5,i_{11}),R(6,i_{12}),R(6,i_{13}),R(7,i_{14}),R(7,i_{15})$；且$i_0,i_1,i_2,i_3,i_4,i_5,i_6,i_7,i_8,i_9,i_{10},i_{11},i_{12},i_{13},i_{14},i_{15}$互不相同，$i_0\%8,i_1\%8,i_2\%8,i_3\%8$互不相同，$i_2\%8,i_3\%8,i_4\%8,i_5\%8$互不相同，$i_4\%8,i_5\%8,i_6\%8,i_7\%8$互不相同，$i_8\%8,i_9\%8,i_{10}\%8,i_{11}\%8$互不相同，$i_{10}\%8,i_{11}\%8,i_{12}\%8,i_{13}\%8$互不相同，$i_{12}\%8,i_{13}\%8,i_{14}\%8,i_{15}\%8$互不相同，$i_0\%8,i_1\%8,i_8\%8,i_9\%8$互不相同，$i_2\%8,i_3\%8,i_{10}\%8,i_{11}\%8$互不相同，$i_4\%8,i_5\%8,i_{12}\%8,i_{13}\%8$互不相同，$i_6\%8,i_7\%8,i_{14}\%8,i_{15}\%8$互不相同，$i_0,i_1,i_2,i_3,i_4,i_5,i_6,i_7,i_8,i_9,i_{10},i_{11},i_{12},i_{13},i_{14},i_{15}\in[0,15]$；此等效为目标符号子集中的 16 个 RS 符号来自 8 个符号子集不同位置的符号，且每 2 个 RS 符号来自同一个符号子集；进一步目标符号子集中来自于由延迟取值差值为 Q*C=144 的两个延迟线输出的符号子集的 4 个 RS 符号位于对应符号子集不同位置；进一步目标符号子集中来自于由卷积交织器同一延迟线输出的 2 个符号子集的 4 个 RS 符号位于对应符号子集不同位置；进一步目标符号子集中最多有 2 个 RS 符号来自于同一个通道数据流，其通过第一分组交织映射到 2 个不同符号子集，且对应的 2 个符号子集由卷积交织器 2 个不同的延迟线输出，且对应 2 个延迟线的延迟差值大于等于 2*Q*C=288 个 RS 符号。由此规则 2 个符号集合交织到 8 个目标符号子集的具体映射有很多种，表 17-24 给出了几种具体的映射关系，每个表第 y 行第 z 列的数字 x 表示目标符号子集 y 的中的第 z 个 RS 符号来自于符号子集$\lfloor x/16\rfloor$的第 x%16 个 RS 符号，其中$0\leq y<8,0\leq z<16,0\leq x<128$。需要说明的是将表中进行任意行之间交换，或者任意列之间交织，或者任意行交换后在进行任意列交换，或者先任意列交换后在进行任意行交换都是一个合法的映射。

根据每个符号子集与通道通道数据流的映射关系，以及符号集合中每个符号子集的时延关系，不难得出目标符号子集与通道数据流的关系为：目标符号子集的数据来源于 K1=8 个不同通道数据流，8 个通道数据流可以表示为通道数据流j_0,通道数据流j_1,通道数据流j_2,通道数据流j_3,通道数据流j_4,通道数据流j_5,通道数据流j_6,通道数据流j_7,且$\lfloor j_x/4 \rfloor = x, x \in [0,7], \lfloor j_x/4 \rfloor$表示对$j_x/4$向下取整；进一步从每个通道数据流取 2 个 RS 符号；此 2 个 RS 符号在对应通道数据流间隔的距离大于等于 a*N*K2/n，即大于等于 2*544*2/32=68 个 RS 符号；进一步最多 2 个 RS 符号来自 2 个不同通道数据流上对齐的 2 个 RS 符号。

经过上述分组交织得到的 32 个目标符号子集分别送入到 32 个内码编码器，每个目标符号子集作为一个内码编码的信息数据，通过内码编码器产生冗余数据，得到 32 条编码数据流。在一种可能的实施方式中，采用 Hamming(170,160)进行内码编码，分别将目标符号子集的 16 个 RS 符号总共 160 比特添加 10 比特冗余得到 170 比特的码字。在另一种可能的实施方式中，采用 BCH(176,160) 进行内码编码，分别将目标符号子集的 16 个 RS 符号总共 160 比特添加 16 比特冗余得到 176 比特的码字。经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。

采用本实施例的数据交织编码方案，当内码采用 Hamming(170,160)时，其 KP4RS(544,514) + Hamming(170,160)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 4.3E-3，性能逼近该级联 FEC 方案的最优性能。当内码采用 BCH(176,160)时，其 KP4 RS(544,514) + BCH(176,160)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 8.3E-3，性能逼近该级联 FEC 方案的最优性能。

实施例 5：内码编码信息位长度为 120 比特，内码并行度 16。

应理解，实施例 1-4 中改变第一分组交织中第三符号矩阵和第四符号矩阵之间的映射关系，将会改变符号子集中 RS 码字符号分布规律，从而将会影响第二分组交织中符号集合和目标符号子集的关系。本实施例 5 给出一种新的第一分组交织方案和对应的第二分组交织具体的实现。

图 49 为第一分组交织的另一种实现方式示意图。如图 49 所示，第四符号矩阵第 i 行第 j 列的符号来自第三符号矩阵的第 x 行第 y 列，其中满足$x = (j\%2) * 4 + i, y = \lfloor j/2 \rfloor, (0 \le i < 4, 0 \le j <$ ） 16),其中$\lfloor j/2 \rfloor$表示为对 j/2向下取整。定义第四符号矩阵一行为一个符号子集，交织后得到的 4 个符号子集分别为第一分组交织输出 T=4 条第一数据流上连续 16 个 RS 符号，并结合图 5 到图 9 所示 PCS/FEC 通道数据流格式，可以得到每个符号子集的符号来自于 8 个不同的通道数据流，8 个通道数据流可以表示为通道数据流j_0,通道数据流j_1,通道数据流j_2,通道数据流j_3,通道数据流j_4,通道数据流j_5,通道数据流j_6,通道数据流j_7,且$\lfloor j_x/4 \rfloor = x, x \in [0,7], \lfloor j_x/4$ 表示对 jx/4 向下取整，至少来自于 4 个不同 RS 码字符号。

经过图 49 所示第一分组处理得到的 4 条第一数据流，再经过图 45（c）所示的卷积交织处理得到 4 条第二数据流。定义第二数据流中从卷积交织器的延迟线 0，延迟线 1、延迟线 2 分别输出的$S_r(3t-6Q)$，$S_r(3t-3Q+1)$，$S_r(3t+2)$三个符号子集为符号集合，其总共包含 48 个 RS 符号。结合图 5 到图 9 中给出各种业务的 PCS/FEC 通道上 RS 码字的分布以及第一分组交

织和卷积交织可以得到，不难理解每个符号子集至少来自于 4 个不同的 RS 码字，且同一符号子集的符号 0 到符号 7 对应的 RS 码字与符号 8 到符号 15 对应的 RS 码字不同；进一步任意 2 个符号子集的符号 0 到符号 7 对应的 RS 码字分布可能一致，任意 2 个符号子集的符号 8 到符号 15 对应的 RS 码字分布可能一致；进一步符号集合至少来自于 12 个不同 RS 码字，且符号集合中最多 4 个符号属于同一个 RS 码字。

因此可以采用如图 45（d）所示的分组交织结构，从第二数据流中获取一个符号集合，从每个符号子集各取 4 个 RS 符号共 12 个 RS 符号映射为一个目标符号子集，总共得到 4 个目标符号子集，4 个目标符号子集分别为分组交织器输出的 4 路目标符号子集上的连续 12 个 RS 符号，即等效为每个分组交织器输出 S=4 条目标符号子集。为表述方便定义$R(x,y)$为符号子集 x 的符号 y（$x \in [0,2], y \in [0,15]$）。为使得目标符号子集中的 12 个 RS 符号来自于 12 个不同 RS 码字，则此 12 个 RS 符号对应于符号集合中的

$R(0,i_0), R(0,i_1), R(0,i_2), R(0,i_3), R(1,i_4), R(1,i_5), R(1,i_6), R(1,i_7)$，$R(2,i_8), R(2,i_9), R(2,i_{10}), R(2,i_{11})$；且满足$i_0, i_1, i_4, i_5, i_8, i_9$互不相同，$i_2, i_3, i_6, i_7, i_{10}, i_{11}$互不相同，$i_0, i_1, i_4, i_5, i_8, i_9 \in [0,7], i_2, i_3, i_6, i_7, i_{10}, i_{11} \in [8,15]$；此等效为目标符号子集中的 12 个 RS 符号来自 3 个符号子集不同位置的符号，且每 4 个符号来自于同一个符号子集，且其中 2 个 RS 符号来自同一个符号子集的符号 0 到符号 7，另外 2 个 RS 符号来自同一个符号子集的符号 8 到符号 15；进一步目标符号子集中来自于由延迟取值差值为 Q*d=176 的两个延迟线输出的符号子集的 8 个 RS 符号位于对应符号子集不同位置；进一步目标符号子集中最多有 2 个 RS 符号来自于同一个通道数据流，其通过第一分组交织映射到 2 个不同符号子集，且对应的 2 个符号子集由 2 个卷积交织器 2 个不同的延迟线输出，且对应 2 个延迟线的延迟差值大于等于 136 个 RS 符号。由此规则符号集合到目标符号子集的具体映射有很多种，表 25-28 给出了几种具体的映射关系，表 25-28 中每个表第 y 行第 z 列的数字 x 表示目标符号子集 y 的中的第 z 个 RS 符号来自于符号子集$\lfloor x/16 \rfloor$的第 x%16 个 RS 符号，其中$0 \le y < 4, 0 \le z < 12, 0 \le x < 48$。需要说明的是将表中进行任意行之间交换，或者任意列之间交换，或者任意行交换后在进行任意列交换，或者先任意列交换后在进行任意行交换都是一个合法的映射。

根据每个符号子集与通道通道数据流的映射关系，以及符号集合中每个符号子集的时延关系，不难得出目标符号子集与通道数据流的关系为：目标符号子集的数据来源于 K1=8 个不同通道数据流，8 个通道数据流可以表示为通道数据流j_0,通道数据流j_1,通道数据流j_2,通道数据流j_3,通道数据流j_4,通道数据流j_5,通道数据流j_6,通道数据流j_7,且$\lfloor j_x/4 \rfloor = x, x \in [0,7]$,$\lfloor j_x/4 \rfloor$表示对$j_x/4$向下取整；进一步从其中 4 个通道数据流取 1 个 RS 符号，从剩下 4 个通道数据流取 2 个 RS 符号；进一步如果从某一个通道数据流取 2 个 RS 符号，这 2 个 RS 符号在对应通道数据流间隔的距离大于 a*N*K2/n，即大于等于 2*544*2/32=68 个 RS 符号；进一步最多 2 个 RS 符号来自 2 个不同通道数据流上对齐的 2 个 RS 符号。

表 25

0	3	8	11	20	23	28	31	34	33	42	41
2	1	10	9	22	21	30	29	32	35	40	43
4	7	12	15	16	19	24	27	38	37	46	45
6	5	14	13	18	17	26	25	36	39	44	47

表 26

0	3	8	11	20	23	28	31	34	33	42	41
2	1	10	9	22	21	30	29	32	35	40	43
4	7	12	15	18	17	26	25	38	37	46	45
6	5	14	13	16	19	24	27	36	39	44	47

表 27

0	3	8	11	22	21	30	29	34	33	42	41
2	1	10	9	20	23	28	31	32	35	40	43
4	7	12	15	18	17	26	25	38	37	46	45
6	5	14	13	16	19	24	27	36	39	44	47

表 28

0	3	8	11	22	21	30	29	34	33	42	41
2	1	10	9	20	23	28	31	32	35	40	43
4	7	12	15	16	19	24	27	38	37	46	45
6	5	14	13	18	17	26	25	36	39	44	47

经过上述分组交织得到的 16 个目标数据流分别送入到内码编码器，每个目标符号子集作为一个内码编码的信息数据，通过内码编码器产生冗余数据，得到 16 条编码数据流。在一种可能的实施方式中，采用 Hamming(128,120)进行内码编码，分别将目标数据流中的每个目标符号子集的 12 个 RS 符号总共 120 比特添加 8 比特冗余得到 128 比特的码字。在另一种可能的实施方式中，采用 BCH(136,120) 进行内码编码，分别将目标数据流中的每个目标符号子集的 12 个 RS 符号总共 120 比特添加 16 比特冗余得到 136 比特的码字。

经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括调制映射（mapping），信道交织（channel interleaving），偏振划分（polarization distribution），或 DSP 成帧处理（Framing）等。比如内码编码数据流可通过交织，以提升系统抗突发错误的能力。

采用本实施例的数据交织编码方案，其 KP4 RS(544,514) + Hamming(128,120)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 4.5E-3，性能逼近该级

联 FEC 方案的最优性能。

图 50（a）为本申请实施例中数据处理装置的另一种结构示意图。如图 50（a）所示，该数据处理装置包括交织模块 301 和编码器 302。交织模块 301 用于执行上述数据处理方法中对 n 条通道数据流进行交织处理得到 m 条目标数据流的操作。编码器 302 用于执行上述数据处理方法中对 m 条目标数据流分别进行第二 FEC 编码得到编码数据流的操作。具体地，交织模块 301 包括卷积交织器 3011 和分组交织器 3012，卷积交织器 3011 用于执行上述图 34 所示实施例中的步骤 3401，分组交织器 3012 用于执行上述图 34 所示实施例中的步骤 3402。

图 50（b）为本申请实施例中数据处理装置的另一种结构示意图。如图 50（b）所示，该数据处理装置包括交织模块 401 和编码器 402。交织模块 401 用于执行上述数据处理方法中对 n 条通道数据流进行交织处理得到 m 条目标数据流的操作。编码器 402 用于执行上述数据处理方法中对 m 条目标数据流分别进行第二 FEC 编码得到编码数据流的操作。具体地，交织模块 401 包括第一分组交织器 4011、卷积交织器 4012 和第二分组交织器 4013，第一分组交织器 4011 用于执行上述图 42 所示实施例中的步骤 4201，卷积交织器 4012 用于执行上述图 42 所示实施例中的步骤 4202，第二分组交织器 4013 用于执行上述图 42 所示实施例中的步骤 4203。

应理解，本申请提供的装置也可以通过其他方式实现。例如，上述装置中的单元划分仅仅是一种逻辑功能划分，实际实现时可以有另外的划分方式，例如，多个单元或组件可以结合或可以集成到另一个系统。另外，本申请各个实施例中的各功能单元可以集成在一个处理单元中，也可以是各个独立的物理单元，也可以是两个或两个以上个功能单元集成在一个处理单元中。上述集成的单元既可以采用硬件的形式实现，也可以采用软件功能单元的形式实现。

需要说明的是，除了上述实施例介绍的数据处理方法外，本申请还提供了另一种数据处理方法，下面进行详细介绍。

图 51 为本申请实施例提供的数据处理方法的另一种流程示意图。

5101、对 n 条通道数据流中每 t 条通道数据流进行分组交织得到 s 条第一数据流，以得到共 m 条第一数据流。

本实施例中，通道数据流可以是 PCS 通道数据流或 FEC 通道数据流，具体此处不做限定。n 条通道数据流都是经过第一 FEC 编码的数据流，也就是上文中介绍的经过外码编码的数据流，其中，n 为大于 1 的整数。例如，该外码编码可以采用 RS 码，经过外码编码后的 n 条数据流可以包括多个 RS 码字。在实际应用中也可以采用其他的编码方式进行外码编码，为了便于描述，下文统一用 RS 码字来表示外码编码后生成的码字。应理解，经过外码编码后的每 a 个码字分布在 b 条通道数据流中，其中，$a \leqslant b \leqslant n$，n 能被 b 整除，a 为大于或等于 1 的整数。在上述图 5-图 9 所示的不同应用场景中，a 和 b 的取值也可能不同。以图 5 所示的应用场景为例，n=32，a=2，b=16，即每 2 个码字分布在 16 条通道数据流中。在图 6-图 9 其他几个应用场景中 a 和 b 的取值可以通过附图推导得出，此处不再一一赘述。需要说明的是，本申请

中外码码长取值是以符号为单位来统计的，其中，符号可以包括一个或多个比特。例如，外码为采用的KP4 RS(544,514)码，码长N=544个符号，一个符号包含10个比特。

具体地，n条通道数据流可以分为q组，每组包括t条通道数据流，其中，q为大于或等于1的整数，n能被q整除。相应的，分组交织后得到的m条第一数据流也可以分为q组，每组包括s条第一数据流。即n=q*t，m=q*s。每组的t条通道数据流经过分组交织后得到对应的一组s条第一数据流。

图52为本申请实施例中对n条通道数据流进行分组交织的一种结构示意图。如图52所示，作为一个示例，由q个分组交织器进行对n条通道数据流进行分组交织，交织器i将通道数据流i*t到通道数据流(i+1)*t-1分组交织得到第一数据流i*s到第一数据流(i+1)*s-1，0≤i＜q。

由于每条通道数据流中每连续a个符号来自a个不同的码字，且每条通道数据流中每连续的L1个符号来自至少a个不同的码字，L1=N*a/b,N表示外码编码得到的码字长度。因此，以其中任一组的t条通道数据流为例，先从每条通道数据流中各取连续的a个符号得到共t*a个符号，再从t*a个符号的每个符号中各取任意的Δ个比特得到共D=Δ*t*a个比特，这D个比特映射到分组交织后任一条第一数据流中连续的D个比特。其中，Δ=M/s，M表示一个符号包含的比特数，为了便于说明，后面都以M=10为例进行介绍。

需要说明的是，经过分组交织后每条第一数据流中每连续的d个符号来自v个不同的码字，且每条第一数据流中每连续的L2个符号来自至少v个不同的码字。其中，v能被a整除，d=D/M，L2=t/s*L1。

在一种可能的实施方式中，当发送设备01发送的1*800GE业务的时候，根据以太网联盟定义的《Ethernet Technology Consortium800G Specification》，发端设备01的RS-FEC采用KP4 RS(544,514)码，称为外码，并将多个RS分配到32个虚拟的PCS通道上，具体如图5所示，PCS通道数据流0-15或PCS通道数据流16-31中每条数据流间隔68个符号总共16*68=1088个符号，其包含了2个RS码码字。每条PCS通道数据流中相邻a=2个符号来自不同RS码码字，且相邻两条PCS通道数据流的同个位置的2个符号来自不同RS码码字，且每连续的L1=68个符号来自至少不同的a=2个RS码字。此32个PCS通道数据流复用后通过通道连接单元接口AUI送到发端处理模块。需要说明的是，2个RS码码字1088个符号分布在16条通道数据流中，所述1088个符号中的68个符号为一条通道数据流中的连续68个符号；当一条通道数据流中连续L1=68个符号来自如图5中的2个虚框，此时该68个符号来自3或4个不同RS码字。为了简单描述，下文简单表述为每条通道数据流中每连续的L1个符号来自a个不同的码字。

基于图3（h）所示的发端处理模块的数据处理示意图，发端处理模块PMA单元将从通道连接单元接口AUI进行de-mux后恢复出32个PCS通道数据流，然后利用PCS通道已知的对齐标识（Alignment marker）进行通道数据流的标识锁定（alignment lock）。这里32个通道的已知对齐标识各不相同（见《Ethernet Technology Consortium 800G Specification》）。然后根据对齐标识对n=32条PCS通道数据流进行通道重排序（lane reorder）处理，使得n=32条PCS通道的数据能够按照指定的顺序排列，一种具体排列方式为通道数据流2*i

（$0 \le i < 16$）共 16 条通道数据流来自相同的 RS 码字，通道数据流 2*i+1（$0 \le i < 16$）共 16 条通道数据流来自相同的 RS 码字。

在另一种可能的实施方式中，当发送设备 01 发送的 1*800GE 业务的时候，基于图 3（h）所示的发端处理模块的数据处理示意图，发端处理模块的 PMA 单元将从通道连接单元接口 AUI 接收的数据进行 de-mux 后恢复出 32 个 PCS 通道数据流，然后利用 PCS 通道已知的对齐标识（Alignment marker）进行通道数据流的标识锁定（alignment lock）。这里 32 个通道的已知对齐标识各不相同（见《Ethernet Technology Consortium 800G Specification》）。发端处理模块接着根据对齐标识对 n=32 条通道的数据进行通道重排序（lane reorder）处理，使得 n=32 条通道数据流能够按照指定的顺序排列，一种具体排列方式为通道数据流 0 到 15 共 16 条通道数据流来自相同的 RS 码字，通道数据流 16 到 31 共 16 条通道数据流来自相同的 RS 码字。需要说明的是，上述“通道数据流 0 到 15 共 16 条通道数据流来自相同的 RS 码字”可参照图 5 进行理解，更具体的是，对于通道数据流 0 到 15，每条通道数据流中的 34 个符号，总共 544 个符号来自相同的 RS 码字；且对于通道数据流 0 到 15，每条通道数据流中的连续 68 个符号，总共 1088 个符号来自 2 个 RS 码字。类似的，对于通道数据流 16 到 31，每条通道数据流中的 34 个符号，总共 544 个符号来自相同的 RS 码字；且对于通道数据流 16 到 31，每条通道数据流中的连续 68 个符号，总共 1088 个符号来自另外 2 个 RS 码字。为了描述简单，本申请会简单表述：32 条通道数据流中通道数据流 0-通道数据流 15 共 16 条通道数据流来自相同的码字，32 条通道数据流中通道数据流 16-通道数据流 31 共 16 条通道数据流来自相同的码字，通道数据流 0-通道数据流 15 和通道数据流 16-通道数据流 31 来自不同的码字。

下面介绍几种分组交织的具体实施方式。

实施方式 1：以 n=32 为例，32 条通道数据流中位于奇数通道的 16 条通道数据流来自相同的码字，32 条通道数据流中位于偶数通道的 16 条通道数据流来自相同的码字，位于奇数通道的数据流和位于偶数通道的数据流来自不同的码字。即通道数据流 2*i 共 16 条通道数据流来自相同的码字，通道数据流 2*i+1 共 16 条通道数据流来自相同的码字，$0 \le i < 16$。

图 53 为本申请实施例中分组交织的一种应用场景示意图。如图 53 所示的 1 个分组交织器用于对 2 条通道数据流进行分组交织得到 1 条第一数据流，分组交织共需要用到 16 个如图 53 所示的分组交织器。具体地，分组交织器 i 对第 2*i 条通道数据流和第 2*i+1 条通道数据流进行分组交织得到 1 条第一数据流，总共得到 m=16 条第一数据流，$0 \le i < 16$。分组交织具体映射方式为：从通道数据流 2*i 和通道数据流 2*i+1 各取 a=2 个符号共 4 个符号，然后以任意顺序映射为第一数据流 i 中连续的 D=40 比特，即 d=4 个符号。其中，分组交织得到的 1 条第一数据流中每连续的 d=4 个符号来自 v=4 个不同的码字，且每连续的 L2=136 个符号来自不同的 v=4 的不同的码字。

一种方式为从通道数据流 2*i 和通道数据流 2*i+1 中各自的连续 20 个比特按照β个比特轮询输出得到第一数据流中连续 d=4 个符号。具体地，令一次分组交织操作得到的第一数据流中的连续 D=40 比特分别表示为 b0~b39，则上述β个比特轮询输出的映射可表示：第一数据

流中连续的 40 个比特中第j个比特来自第$2*i+\left\lfloor\frac{j}{\beta}\right\rfloor\%2$条通道数据流中连续 20 个比特中的第$\left\lfloor\left\lfloor\frac{j}{\beta}\right\rfloor/2\right\rfloor*\beta+j\%\beta$个比特，其中，$\beta$可以为 20 的约数，即$\beta=1,2,4,5,10,20$，

[x]表示向下取整，$0\le j<40$。

图 54 为本申请实施例中分组交织的几种具体实施例示意图。如图 54 的（a）示例表示$\beta=10$时分组交织的实施例，如图 54 的（b）示例表示$\beta=20$时分组交织的实施例，如图 54 的（c）示例表示$\beta=1$时分组交织的实施例。

实施方式 2：以 n=32 为例，32 条通道数据流中位于奇数通道的 16 条通道数据流来自相同的码字，32 条通道数据流中位于偶数通道的 16 条通道数据流来自相同的码字，位于奇数通道的数据流和位于偶数通道的数据流来自不同的码字。即通道数据流 2*i 共 16 条通道数据流来自相同的码字，通道数据流 2*i+1 共 16 条通道数据流来自相同的码字，$0\le i<16$。

图 55 为本申请实施例中分组交织的另一种应用场景示意图。如图 55 所示的 1 个分组交织器用于对 2 条通道数据流进行分组交织得到 2 条第一数据流，分组交织共需要用到 16 个如图 55 所示的分组交织器。具体地，分组交织器 i 对第 2*i 条通道数据流和第 2*i+1 条通道数据流进行分组交织得到第 2*i 条第一数据流和第 2*i+1 条第一数据流，0≤i＜16。分组交织具体映射方式为：从通道数据流 2*i 和通道数据流 2*i+1 各获取连续的 a=2 个符号共 4 个符号，从每个符号中各任取 Δ =M/s=10/2=5 个比特共 D=20 比特以任意顺序映射为第一数据流 2*i 或者第二数据流 2*i+1 中连续的 D=20 比特,即 d=2 符号。其中，分组交织后第一数据流中每连续 D=20 比特来自 v=4 个不同的码字，进一步每个第一数据流中每 L2=68 符号来自不同的 4 个码字。

图 56 为本申请实施例中分组交织的几种具体实施例示意图。用B_f^{2*i+g}表示分组交织 i 分组一次分组交织操作生成的第一数据流 2*i+g 的连续的 20bits 中的第 f 比特。如图 56 的(a)示例所示，B_f^{2*i+g}来自第$2*i+\left\lfloor\frac{f}{10}\right\rfloor$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f\%10}{5}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。如图 56 的（b）示例所示，$B_f^{2*i+g}$来自第$2*i+\left\lfloor\frac{f}{5}\right\rfloor\%2$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f}{10}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

实施方式 3：以 n=32 为例，32 条通道数据流中位于奇数通道的 16 条通道数据流来自相同的码字，32 条通道数据流中位于偶数通道的 16 条通道数据流来自相同的码字，位于奇数通道的数据流和位于偶数通道的数据流来自不同的码字。即通道数据流 2*i 共 16 条通道数据流来自相同的码字，通道数据流 2*i+1 共 16 条通道数据流来自相同的码字，$0\le i<16$。

图 57 为本申请实施例中分组交织的另一种应用场景示意图。在该应用场景中，分组交织具体采用复用的形式，如图 57 所示的 1 个复用器用于对 2 条通道数据流进行复用得到 1 条第一数据流，复用共需要用到 16 个如图 57 所示的复用器。具体地，复用器 i 对第 2*i 条通道数据流和第 2*i+1 条通道数据流进行复用得到 1 条第一数据流，0≤i＜16。复用具体映射方式为：C_{2i}^{j}表示第 2*i 条通道数据流中第 j 组连续的 β 个比特，C_{2i+1}^{j}表示第 2*i+1 条通道数据流中第 j 组连续的 β 个比特，$j\ge0$，C_{2i}^{j}和C_{2i+1}^{j}在复用得到的第一数据流中连续。即经过

2：1 复用后第一数据流数据顺序为C_{2i}^{0}，C_{2i+1}^{0},C_{2i}^{1}，C_{2i+1}^{1},…,C_{2i}^{w}，C_{2i+1}^{w},…，其中$0 \le i \le 15$，β =1，2，4，5，10，20。例如，当β =1 时表示第 2*i 条通道数据流和第 2*i+1 条通道数据流采用比特复用（bit-mux）得到第一数据流 i。又例如，当β =10 时表示第 2*i 条通道数据流和第 2*i+1 条通道数据流采用 RS 符号复用（symbol-mux）得到第一数据流 i。再例如，当β =20 时表示第 2*i 条通道数据流和第 2*i+1 条通道数据流采用 2RS 符号复用（symbol-mux）得到第一数据流 i。经过 2：1 复用后，第一数据流中每连续的 d=4 个符号来自 4 个不同的 RS 码字。

实施方式 4：以 n=32 为例，32 条通道数据流中通道数据流 0-通道数据流 15 共 16 条通道数据流来自相同的码字，32 条通道数据流中通道数据流 16-通道数据流 31 共 16 条通道数据流来自相同的码字，通道数据流 0-通道数据流 15 和通道数据流 16-通道数据流 31 来自不同的码字。

图 58 为本申请实施例中分组交织的另一种应用场景示意图。如图 58 所示的 1 个分组交织器用于对 2 条通道数据流进行分组交织得到 1 条第一数据流，分组交织共需要用到 16 个如图 58 所示的分组交织器。具体地，分组交织器 i 对第 i 条通道数据流和第 i+16 条通道数据流进行分组交织得到 1 条第一数据流，总共得到 m=16 条第一数据流，$0 \le i < 16$。分组交织具体映射方式为：从通道数据流 i 和通道数据流 i+16 各取 a=2 个符号共 4 个符号，然后以任意顺序映射为第一数据流 i 中连续的 D=40 比特，即 d=4 个符号。其中，分组交织得到的 1 条第一数据流中每连续的 d=4 个符号来自 v=4 个不同的码字，且每连续的 L2=136 个符号来自不同的 v=4 的不同的码字。

一种方式为通道数据流 i 和通道数据流 i+16 中各自的 2 个连续符号按照β个比特轮询输出得到第一数据流中连续 d=4 个符号。具体地，令一次分组交织操作得到的第一数据流中的连续 D=40 比特分别表示为 b0~b39。则上述β个比特轮询输出的映射可表示：第一数据流中连续的 40 个比特中第j个比特来自第$2*i+\left\lfloor \frac{j}{\beta} \right\rfloor \%2$条通道数据流中连续 20 个比特中的第$\left\lfloor \left\lfloor \frac{j}{\beta} \right\rfloor /2 \right\rfloor * \beta + j\%\beta$个比特，所述$\lfloor x \rfloor$表示向下取整，$0 \le j < 40$，所述$\beta$为 1,2,4,5,10 或 20。

实施方式 5：以 n=32 为例，32 条通道数据流中通道数据流 0-通道数据流 15 共 16 条通道数据流来自相同的码字，32 条通道数据流中通道数据流 16-通道数据流 31 共 16 条通道数据流来自相同的码字，通道数据流 0-通道数据流 15 和通道数据流 16-通道数据流 31 来自不同的码字。

图 59 为本申请实施例中分组交织的另一种应用场景示意图。如图 59 所示的 1 个分组交织器用于对 2 条通道数据流进行分组交织得到 2 条第一数据流，分组交织共需要用到 16 个如图 59 所示的分组交织器。具体地，分组交织器 i 对第 i 条通道数据流和第 i+16 条通道数据流进行分组交织得到第 2*i 条第一数据流和第 2*i+1 条第一数据流，$0 \le i < 16$。分组交织具体映射方式为从通道数据流 i 和通道数据流 i+16 各取 a=2 个 RS 符号共 4 个 RS 符号，从每个符号中各任取Δ =M/s=10/2=5 比特数据共 D=20 比特以任意顺序映射为第一数据流 2*i 或者第二数据流 2*i+1 中连续的 D=20 比特，即 d=2 符号。其中，分组交织后第一数据流中每连续 D=20

比特来自 v=4 个不同的码字，进一步每个第一数据流中每 L2=68 符号来自不同的 4 个码字。

在一种可能的实施方式中，用B_f^{2*i+g}表示分组交织 i 分组交织操作生成了第一数据流 2*i+g 的连续的 20bits 中的第 f 比特。B_f^{2*i+g}来自第$i+16*\left\lfloor\frac{f}{10}\right\rfloor$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f\%10}{5}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

在一种可能的实施方式中，用B_f^{2*i+g}表示分组交织 i 分组交织操作生成了第一数据流 2*i+g 的连续的 20bits 中的第 f 比特。B_f^{2*i+g}来自第$i+16*\left\lfloor\frac{f}{10}\right\rfloor$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f}{10}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

实施方式 6：以 n=32 为例，32 条通道数据流中通道数据流 0-通道数据流 15 共 16 条通道数据流来自相同的码字，32 条通道数据流中通道数据流 16-通道数据流 31 共 16 条通道数据流来自相同的码字，通道数据流 0-通道数据流 15 和通道数据流 16-通道数据流 31 来自不同的码字。

图 60 为本申请实施例中分组交织的另一种应用场景示意图。在该应用场景中，分组交织具体采用复用的形式，如图 60 所示的 1 个复用器用于对 2 条通道数据流进行复用得到 1 条第一数据流，复用共需要用到 16 个如图 60 所示的复用器。具体地，复用器 i 对第 i 条通道数据流和第 i+16 条通道数据流进行复用得到 1 条第一数据流，$0\le i<16$。复用具体映射方式为：C_i^j表示第 i 条通道数据流中第 j 组连续的 β 个比特，C_{i+16}^j表示第 i+16 条通道数据流中第 j 组连续的 β 个比特，$j\ge0$，C_i^j和C_{i+16}^j在复用得到的第一数据流中连续。即经过 2：1 复用后第一数据流数据顺序为C_i^0，C_{i+16}^0，C_i^1，C_{i+16}^1，…，C_i^w，C_{i+16}^w，…，其中$0\le i\le15$，β =1，2，4，5，10，20。例如，当 β =1 时表示第 i 条通道数据流和第 i+16 条通道数据流采用比特复用（bit-mux）得到第一数据流 i。又例如，当 β =10 时表示第 i 条通道数据流和第 i+16 条通道数据流采用 RS 符号复用（symbol-mux）得到第一数据流 i。再例如，当 β =20 时表示第 i 条通道数据流和第 i+16 条通道数据流采用 2RS 符号复用（symbol-mux）得到第一数据流 i。经过 2：1 复用后，第一数据流中每连续的 d=4 个符号来自 4 个不同的 RS 码字。

需要说明的是，复用器 i 对第 i 条通道数据流和第 i+16 条通道数据流进行复用得到 1 条第一数据流，也即复用器 i 的两条输入数据流为第 i 条通道数据流和第 i+16 条通道数据流，也可以是其他具体实现方式。一种具体实现方式是复用器 i 对第 i 条通道数据流和第 31-i 条通道数据流进行复用得到 1 条第一数据流。需要说明的是，任意一个复用器的 2 条输入数据流满足如下约束即可：其中一条输入数据流来自通道数据流 0-通道数据流 15 中的一条，另一条输入数据流来自通道数据流 16-通道数据流 31 中的一条。其具体实现方式可根据上述实施例进行简单扩展，其具体实现方式本领域普通技术人员可知，此处不再赘述。

实施方式 7：以 n=32 为例，32 条通道数据流中通道数据流 0-通道数据流 15 共 16 条通道数据流来自相同的码字，32 条通道数据流中通道数据流 16-通道数据流 31 共 16 条通道数据流来自相同的码字，通道数据流 0-通道数据流 15 和通道数据流 16-通道数据流 31 来自不同的码字。更具体的是，参照图 5 进行理解，对于通道数据流 0 到 15，每条通道数据流中的

34 个符号，总共 544 个符号来自相同的 RS 码字；且对于通道数据流 0 到 15，每条通道数据流中的连续 68 个符号，总共 1088 个符号来自 2 个 RS 码字。类似的，对于通道数据流 16 到 31，每条通道数据流中的 34 个符号，总共 544 个符号来自相同的 RS 码字；且对于通道数据流 16 到 31，每条通道数据流中的连续 68 个符号，总共 1088 个符号来自另外 2 个 RS 码字。

图 61 为本申请实施例中分组交织的再一种应用场景示意图。如图 61 所示的 1 个分组交织器用于对 4 条通道数据流进行分组交织得到 1 条第一数据流，分组交织共需要用到 8 个如图 61 所示的分组交织器。其中，所述 4 条通道数据流中，2 条通道数据流来自通道数据流 0-通道数据流 15 中的两条，另 2 条通道数据流来自通道数据流 16-通道数据流 31 中的两条。分组交织器从所述 4 条通道数据流各取 a=2 个连续的符号共 8 个符号，然后映射为所述第一数据流中连续的 D=80 个比特，即 d=8 个符号。其中，分组交织得到的 1 条第一数据流中上述每连续的 d=8 个符号来自至少 v=4 个不同的码字；且每连续的 L2=272 个符号来自至少 v=4 的不同的码字。进一步有分组交织得到的 1 条第一数据流中上述每连续的 d=8 个符号中的每连续的 4 个符号来自 4 个不同的 RS 码字，即每连续 8 个符号中的第 0 个符号到第 3 个符号分别来自不同的 RS 码字，第 4 个符号到第 7 个符号分别来自不同的 RS 码字。一种具体实现方式是，分组交织器 i 对第 2*i 条通道数据流、第 2*i+1 条通道数据流、第 2*i+16 条通道数据流、第 2*i+17 条通道数据流进行分组交织得到 1 条第一数据流，总共得到 m=8 条第一数据流，0≤i≤7。

分组交织的一种具体交织方式为：从第 2*i 条通道数据流、第 2*i+1 条通道数据流、第 2*i+16 条通道数据流、第 2*i+17 条通道数据流（0≤i＜8）各取 a=2 个连续的符号共 8 个符号，然后交织为第一数据流 i 中连续的 D=80 个比特，即 d=8 个符号。一种具体实施方式为，上述第一数据流 i 中连续的 8 个符号中，第 0 和第 1 个符合来自第 2*i 条通道数据流，第 2 和第 3 个符号来自第 2*i+16 条通道数据流，第 4 和第 5 个符号来自第 2*i+1 条通道数据流，第 6 和第 7 个符号来自第 2*i+17 条通道数据流。

图 62 为本申请实施例中分组交织的再一种应用场景示意图。需要说明的是，所述分组交织也可采用如图 62 所示的符号复用（symbol-mux）实现，其中C_{2*i}^{j}表示第$2*i$条通道数据流中第j组连续的 2 个符号，C_{2*i+16}^{j}表示第$2*i+16$条通道数据流中第j组连续的 2 个符号，C_{2*i+1}^{j}表示第$2*i+1$条通道数据流中第j组连续的 2 个符号，C_{2*i+17}^{j}表示第$2*i+17$条通道数据流中第j组连续的 2 个符号，$j\geq 0$，且C_{2*i}^{j}、C_{2*i+16}^{j}、C_{2*i+1}^{j}和C_{2*i+17}^{j}在复用得到的第一数据流中连续。即经过 4：1 符号复用后第一数据流i的数据顺序为

$C_{2*i}^{0},C_{2*i+16}^{0},C_{2*i+1}^{0},C_{2*i+17}^{0},\ldots,C_{2*i}^{j},C_{2*i+16}^{j},C_{2*i+1}^{j},C_{2*i+17}^{j},\ldots$，其中$0\leq i\leq 7$。

需要说明的是，上述 4:1 符号复用（symbol-mux）也可以采用多级符号复用进行实现，如先将第 2*i 条通道数据流和第 2*i+16 条通道数据流进行复用得到第一复用数据流，第 2*i+1 条通道数据流和第 2*i+17 条通道数据流进行 2:1 符号复用得到第二复用数据流，然后将上述第一复用数据流和第一复用数据流进行 2:1 符号复用得到第一数据流i。

实施方式 8：在实施方式 7 基础上，本实施方式 8 提供了分组交织的另一种具体实现方

式。

从第 2*i 条通道数据流、第 2*i+1 条通道数据流、第 2*i+16 条通道数据流、第 2*i+17 条通道数据流（0≤i＜8）各获取 a=2 个连续的符号共 8 个符号，然后交织为第 i 数据流 i 中连续的 D=80 个比特，即 d=8 个符号。在一些具体应用场景下，通道数据对齐处理是基于 2 个 RS 符号对齐处理。此时，上述第 2*i 条通道数据流所获取的 2 个连续符号中的第 0 个符号、第 2*i+1 条通道数据流所获取的 2 个连续符号中的第 0 个符号来自不同 RS 码字，第 2*i 条通道数据流所获取的 2 个连续符号中的第 1 个符号、第 2*i+1 条通道数据流所获取的 2 个连续符号中的第 1 个符号来自不同 RS 码字；上述第 2*i+16 条通道数据流所获取的 2 个连续符号中的第 0 个符号、第 2*i+17 条通道数据流所获取的 2 个连续符号中的第 0 个符号来自不同 RS 码字，第 2*i+16 条通道数据流所获取的 2 个连续符号中的第 1 个符号、第 2*i+17 条通道数据流所获取的 2 个连续符号中的第 1 个符号来自不同 RS 码字。

图 63 为本申请实施例中分组交织的再一种应用场景示意图。结合上述特征，分组交织的另一种具体实施方式如图 63 的（a）示例所示，上述的第一数据流 i 中连续的 8 个符号中，第 0 和第 4 个符合来自第 2*i 条通道数据流，第 1 和第 5 个符号来自第 2*i+16 条通道数据流，第 2 和第 6 个符号来自第 2*i+1 条通道数据流，第 3 和第 7 个符号来自第 2*i+17 条通道数据流。需要说明的是，上述实现方式，等价于，从所述 4 条通道数据流中分别获取 1 个符号，进行交织得到数据流 i 中的连续 4 个符号，其中所述的第一数据流 i 中连续的 4 个符号中，第 0 个符号来自第 2*i 条通道数据流，第 1 个符号来自第 2*i+16 条通道数据流，第 2 个符号来自第 2*i+1 条通道数据流，第 3 个符号来自第 2*i+17 条通道数据流。

分组交织的再一种具体实施方式如图 63 的（b）示例所示，上述的第一数据流 i 中连续的 8 个符号中，第 0 和第 4 个符号来自第 2*i 条通道数据流，第 1 和第 5 个符号来自第 2*i+1 条通道数据流，第 2 和第 6 个符号来自第 2*i+16 条通道数据流，第 3 和第 7 个符号来自第 2*i+17 条通道数据流。需要说明的是，上述实现方式，等价于，从所述 4 条通道数据流中分别获取 1 个符号，进行交织得到数据流 i 中的连续 4 个符号，其中所述的第一数据流 i 中连续的 4 个符号中，第 0 个符合来自第 2*i 条通道数据流，第 1 个符号来自第 2*i+1 条通道数据流，第 2 个符号来自第 2*i+16 条通道数据流，第 3 个符号来自第 2*i+17 条通道数据流。

需要说明的是，所述分组交织也可采用符号复用（symbol-mux）实现。符号复用从 4 条通道数据流中分别获取 1 个符号，其中C_{2*i}^{j}表示第$2*i$条通道数据流中第j个符号，C_{2*i+16}^{j}表示第$2*i+16$条通道数据流中第j个符号，C_{2*i+1}^{j}表示第$2*i+1$条通道数据流中第j个符号，C_{2*i+17}^{j}表示第$2*i+17$条通道数据流中第j个符号，$j\geq 0$，且C_{2*i}^{j}、C_{2*i+16}^{j}、C_{2*i+1}^{j}和C_{2*i+17}^{j}在 4:1 符号复用得到的第一数据流中连续。即经过 4：1 符号复用后第一数据流i的数据顺序为$C_{2*i}^{0},C_{2*i+16}^{0},C_{2*i+1}^{0},C_{2*i+17}^{0},\ldots,C_{2*i}^{j},C_{2*i+16}^{j},C_{2*i+1}^{j},C_{2*i+17}^{j},\ldots$，其中$0\leq i\leq 7$。其中$C_{2*i}^{j}$、$C_{2*i+16}^{j}$、$C_{2*i+1}^{j}$和$C_{2*i+17}^{j}$来自 4 个不同的 RS 码字。另一种具体的符号复用是符号复用从 4 条通道数据流中分别获取 1 个符号，C_{2*i}^{j}、C_{2*i+1}^{j}、C_{2*i+16}^{j}和C_{2*i+17}^{j}在 4:1 符号复用得到的第一数据流中连续。即经过 4：1 符号复用后第一数据流i的数据顺序为$C_{2*i}^{0},C_{2*i+1}^{0},C_{2*i+16}^{0},C_{2*i+17}^{0},\ldots,C_{2*i}^{j},C_{2*i+1}^{j},C_{2*i+16}^{j},C_{2*i+17}^{j},\ldots$，其中$0\leq i\leq 7$。

实施方式 9：以 n=32 为例，32 条通道数据流中位于奇数通道的 16 条通道数据流来自相

同的码字，32 条通道数据流中位于偶数通道的 16 条通道数据流来自相同的码字，位于奇数通道的数据流和位于偶数通道的数据流来自不同的码字。即通道数据流 2*i 共 16 条通道数据流来自相同的码字，通道数据流 2*i+1 共 16 条通道数据流来自相同的码字，$0 \leq i < 16$。

图 64 为本申请实施例中分组交织的再一种应用场景示意图。如图 64 所示的 1 个分组交织器用于对 4 条通道数据流进行分组交织得到 1 条第一数据流，分组交织共需要用到 8 个如图 64 所示的分组交织器。其中，所述 4 条通道数据流中，2 条通道数据流来自通道数据流 0-通道数据流 31 中的两条奇数通道数据流，另 2 条通道数据流来自通道数据流 0-通道数据流 31 中的两条偶数通道数据流。分组交织器从所述 4 条通道数据流各取 a=2 个连续的符号共 8 个符号，然后映射为所述第一数据流中连续的 D=80 个比特，即 d=8 个符号。其中，分组交织得到的 1 条第一数据流中上述每连续的 d=8 个符号来自至少 v=4 个不同的码字；且每连续的 L2=272 个符号来自至少 v=4 个不同的码字。进一步有分组交织得到的 1 条第一数据流中上述每连续的 d=8 个符号中的每连续的 4 个符号来自 4 个不同的 RS 码字，即每连续 8 个符号中的第 0 个符号到第 3 个符号分别来自不同的 RS 码字，第 4 个符号到第 7 个符号分别来自不同的 RS 码字。一种具体实现方式是，分组交织器 i 对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流进行分组交织得到 1 条第一数据流，总共得到 m=8 条第一数据流，0≤i≤7。

分组交织的一种具体交织方式为：从第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流（0≤i＜8）各取 a=2 个连续的符号共 8 个符号，然后交织为第一数据流 i 中连续的 D=80 个比特，即 d=8 个符号。一种具体实施方式为，上述第一数据流 i 中连续的 8 个符号中，第 0 和第 1 个符合来自第 4*i 条通道数据流，第 2 和第 3 个符号来自第 4*i+1 条通道数据流，第 4 和第 5 个符号来自第 4*i+2 条通道数据流，第 6 和第 7 个符号来自第 4*i+3 条通道数据流。

图 65 为本申请实施例中分组交织的再一种应用场景示意图。需要说明的是，所述分组交织也可采用如图 65 所示的符号复用（symbol-mux）实现，其中C_{4*i}^{j}表示第$4*i$条通道数据流中第j组连续的 2 个符号，C_{4*i+1}^{j}表示第$4*i+1$条通道数据流中第j组连续的 2 个符号，C_{4*i+2}^{j}表示第$4*i$+2 条通道数据流中第j组连续的 2 个符号，C_{4*i+3}^{j}表示第$4*i+3$条通道数据流中第j组连续的 2 个符号，j≥0，且C_{4*i}^{j}、C_{4*i+1}^{j}、C_{4*i+2}^{j}、和C_{4*i+3}^{j}在复用得到的第一数据流中连续。即经过 4：1 符号复用后第一数据流i的数据顺序为

$C_{4*i}^{0}, C_{4*i+1}^{0}, C_{4*i+2}^{0}, C_{4*i+3}^{0}, \ldots, C_{4*i}^{j}, C_{4*i+1}^{j}, C_{4*i+2}^{j}, C_{4*i+3}^{j}, \ldots$，其中$0 \leq i \leq 7$。

需要说明的是，上述 4:1 符号复用（symbol-mux）也可以采用多级符号复用进行实现，如先将第 4*i 条通道数据流和第 4*i+1 条通道数据流进行复用得到第一复用数据流，第 4*i+2 条通道数据流和第 4*i+3 条通道数据流进行 2:1 符号复用得到第二复用数据流，然后将上述第一复用数据流和第一复用数据流进行 2:1 符号复用得到第一数据流i。

实施方式 10：在实施方式 9 基础上，本实施方式 10 提供了分组交织的另一种具体实现

方式。

从第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流（0≤i＜8）各获取 a=2 个连续的符号共 8 个符号，然后交织为第 i 数据流 i 中连续的 D=80 个比特，即 d=8 个符号。在一些具体应用场景下，通道数据对齐处理是基于 2 个 RS 符号对齐处理。此时，上述第 4*i 条通道数据流所获取的 2 个连续符号中的第 0 个符号与第 4*i+2 条通道数据流所获取的 2 个连续符号中的第 0 个符号来自不同 RS 码字，第 4*i 条通道数据流所获取的 2 个连续符号中的第 1 个符号与第 4*i+2 条通道数据流所获取的 2 个连续符号中的第 1 个符号来自不同 RS 码字；上述第 4*i+1 条通道数据流所获取的 2 个连续符号中的第 0 个符号与第 4*i+3 条通道数据流所获取的 2 个连续符号中的第 0 个符号来自不同 RS 码字，第 4*i+1 条通道数据流所获取的 2 个连续符号中的第 1 个符号与第 4*i+3 条通道数据流所获取的 2 个连续符号中的第 1 个符号来自不同 RS 码字。

图 66 为本申请实施例中分组交织的再一种应用场景示意图。结合上述特征，分组交织的另一种具体实施方式如图 66 的（a）示例所示，上述的第一数据流 i 中连续的 8 个符号中，第 0 和第 4 个符合来自第 4*i 条通道数据流，第 1 和第 5 个符号来自第 4*i+1 条通道数据流，第 2 和第 6 个符号来自第 4*i+2 条通道数据流，第 3 和第 7 个符号来自第 4*i+3 条通道数据流。需要说明的是，上述实现方式，等价于，从所述 4 条通道数据流中分别获取 1 个符号，进行交织得到数据流 i 中的连续 4 个符号，其中所述的第一数据流 i 中连续的 4 个符号中，第 0 个符号来自第 4*i 条通道数据流，第 1 个符号来自第 4*i+1 条通道数据流，第 2 个符号来自第 4*i+2 条通道数据流，第 3 个符号来自第 4*i+3 条通道数据流。

分组交织的再一种具体实施方式如图 66 的（b）示例所示，上述的第一数据流 i 中连续的 8 个符号中，第 0 和第 4 个符号来自第 4*i 条通道数据流，第 1 和第 5 个符号来自第 4*i +2 条通道数据流，第 2 和第 6 个符号来自第 4*i+1 条通道数据流，第 3 和第 7 个符号来自第 4*i+3 条通道数据流。需要说明的是，上述实现方式，等价于，从所述 4 条通道数据流中分别获取 1 个符号，进行交织得到数据流 i 中的连续 4 个符号，其中所述的第一数据流 i 中连续的 4 个符号中，第 0 个符合来自第 4*i 条通道数据流，第 1 个符号来自第 4*i+2 条通道数据流，第 2 个符号来自第 4*i +1 条通道数据流，第 3 个符号来自第 4*i +3 条通道数据流。

需要说明的是，所述分组交织也可采用符号复用（symbol-mux）实现。符号复用从 4 条通道数据流中分别获取 1 个符号，其中C_{4*i}^{j}表示第$4*i$条通道数据流中第j个符号，C_{4*i+1}^{j}表示第$4*i$+1 条通道数据流中第j个符号，C_{4*i+2}^{j}表示第$4*i+2$条通道数据流中第j个符号，C_{4*i+3}^{j}表示第$4*i+3$条通道数据流中第j个符号，$j\geq 0$，且C_{4*i}^{j}、C_{4*i+1}^{j}、C_{4*i+2}^{j}和C_{4*i+3}^{j}在 4:1 符号复用得到的第一数据流中连续。即经过 4：1 符号复用后第一数据流i的数据顺序为 $C_{4*i}^{0},C_{4*i+1}^{0},C_{4*i+2}^{0},C_{4*i+3}^{0},\ldots,C_{4*i}^{j},C_{4*i+1}^{j},C_{4*i+2}^{j},C_{4*i+3}^{j},\ldots$，其中$0\leq i\leq 7$。其中$C_{4*i}^{j}$、$C_{4*i+1}^{j}$、$C_{4*i+2}^{j}$和$C_{4*i+3}^{j}$来自 4 个不同的 RS 码字。另一种具体的符号复用是符号复用从 4 条通道数据流中分别获取 1 个符号，C_{4*i}^{j}、C_{4*i+2}^{j}、C_{4*i+1}^{j}和C_{4*i+3}^{j}在 4:1 符号复用得到的第一数据流中连续。即经过 4：1 符号复用后第一数据流i的数据顺序为 $C_{4*i}^{0},C_{4*i+2}^{0},C_{4*i+1}^{0},C_{4*i+3}^{0},\ldots,C_{4*i}^{j},C_{4*i+2}^{j},C_{4*i+1}^{j},C_{4*i+3}^{j},\ldots$，其中$0\leq i\leq 7$。

需要说明的是，在一些具体实施场景下，发送设备 01 发送的 1*800GE 业务的时候，32 个 PCS 通道数据流在进行 PMA4:1 复用后通过通道连接单元接口 AUI 送到发端处理模块。在进行上述 PMA4：1 复用时，4 条输入数据流中 2 条来自第 0-15 条 PCS 通道数据流中的 2 条，另 2 条来自第 16-31 条 PCS 通道数据流中的 2 条。在发端处理模块的数据处理时，可结合上述“PMA4：1 复用时，4 条输入数据流中 2 条来自第 0-15 条 PCS 通道数据流中的 2 条，另 2 条来自第 16-31 条 PCS 通道数据流中的 2 条”特征，先进行 de-mux 后恢复出 32 个 PCS 通道数据流，其中每条物理通道数据流解复用 de-mux 得到 4 条 PCS 通道数据流，然后利用 PCS 通道已知的对齐标识（Alignment marker）进行通道数据流的标识锁定（alignment lock）；接着不需进行通道重排序，直接将上述物理通道数据流解复用 de-mux 得到的 4 条 PCS 通道数据流进行实施方式 7 或实施方式 8 或实施方式 9 或实施方式 10 所述的分组交织(或符号复用)，将 4 条通道数据流进行分组交织（或符号复用）得到 1 条第一数据流，使得所述第一数据流中连续的 8 个符号来自至少 4 个 RS 码字；其具体实现方式可参照图 61 或图 62 或图 63 或图 64 或图 65 或图 66，其具体实现方式本领域普通技术人员可知，此处不再赘述。

实施方式 11：以 n=32 为例，32 条通道数据流中通道数据流 0-通道数据流 15 共 16 条通道数据流来自相同的码字，32 条通道数据流中通道数据流 16-通道数据流 31 共 16 条通道数据流来自相同的码字，通道数据流 0-通道数据流 15 和通道数据流 16-通道数据流 31 来自不同的码字。更具体的是，参照图 5 进行理解，对于通道数据流 0 到 15，每条通道数据流中的 34 个符号，总共 544 个符号来自相同的 RS 码字；且对于通道数据流 0 到 15，每条通道数据流中的连续 68 个符号，总共 1088 个符号来自 2 个 RS 码字。类似的，对于通道数据流 16 到 31，每条通道数据流中的 34 个符号，总共 544 个符号来自相同的 RS 码字；且对于通道数据流 16 到 31，每条通道数据流中的连续 68 个符号，总共 1088 个符号来自另外 2 个 RS 码字。

图 67 为本申请实施例中分组交织的再一种应用场景示意图。如图 67 所示的 1 个分组交织器用于对 8 条通道数据流进行分组交织得到 1 条第一数据流，分组交织共需要用到 4 个如图 67 所示的分组交织器。其中，所述 8 条通道数据流中，4 条通道数据流来自通道数据流 0-通道数据流 15 中的 4 条，另 4 条通道数据流来自通道数据流 16-通道数据流 31 中的 4 条。分组交织器从所述 8 条通道数据流各取 a=2 个连续的符号共 16 个符号，然后映射为所述第一数据流中连续的 D=160 个比特，即 d=16 个符号。其中，分组交织得到的 1 条第一数据流中每连续的 d=16 个符号来自至少 v=4 个不同的码字，且每连续的 L2=544 个符号来自至少 v=4 的不同的码字。进一步有分组交织得到的 1 条第一数据流中上述每连续的 d=16 个符号中的每连续的 4 个符号来自 4 个不同的 RS 码字,即每连续 16 个符号中的第 0 个符号到第 3 个符号分别来自不同的 RS 码字，第 4 个符号到第 7 个符号分别来自不同的 RS 码字，第 8 个符号到第 11 个符号分别来自不同的 RS 码字，第 12 个符号到第 15 个符号分别来自不同的 RS 码字。一种具体实现方式是，分组交织器 i 对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流、第 4*i+16 条通道数据流、第 4*i+17 条通道数据流、第 4*i+18 条通道数据流、第 4*i+19 条通道数据流进行分组交织得到 1 条第一数据流，总共

得到 m=4 条第一数据流， 0≤i≤3。

分组交织的一种具体交织方式为：从第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流、第 4*i+16 条通道数据流、第 4*i+17 条通道数据流、第 4*i+18 条通道数据流、第 4*i+19 条通道数据流（0≤i＜3）各获取 a=2 个连续的符号共 16 个符号，然后交织为第一数据流 i 中连续的 16 个符号。一种具体实施方式为，第一数据流 i 中连续的 16 个符号中，第 0 和第 1 个符号来自第 4*i 条通道数据流，第 2 和第 3 个符号来自第 4*i+16 条通道数据流，第 4 和第 5 个符号来自第 4*i+1 条通道数据流，第 6 和第 7 个符号来自第 4*i+17 条通道数据流，第 8 和第 9 个符号来自第 4*i+2 条通道数据流，第 10 和第 11 个符号来自第 4*i+18 条通道数据流，第 12 和第 13 个符号来自第 4*i+3 条通道数据流，第 14 和第 15 个符号来自第 4*i+19 条通道数据流。

图 68 为本申请实施例中分组交织的再一种应用场景示意图。需要说明的是，所述分组交织也可采用如图 68 所示的符号复用（symbol-mux）实现，其中C_{4*i}^{j}表示第$4*i$条通道数据流中第j组连续的 2 个符号，C_{4*i+16}^{j}表示第$4*i$+16 条通道数据流中第j组连续的 2 个符号，C_{4*i+1}^{j}表示第$4*i+1$条通道数据流中第j组连续的 2 个符号，C_{4*i+17}^{j}表示第$4*i+17$条通道数据流中第j组连续的 2 个符号，C_{4*i+2}^{j}表示第$4*i$+2 条通道数据流中第j组连续的 2 个符号，C_{4*i+18}^{j}表示第$4*i+18$条通道数据流中第j组连续的 2 个符号，C_{4*i+3}^{j}表示第$4*i+3$条通道数据流中第j组连续的 2 个符号，C_{4*i+19}^{j}表示第$4*i+19$条通道数据流中第j组连续的 2 个符号，$j\geq 0$，且C_{4*i}^{j}、C_{4*i+16}^{j}、C_{4*i+1}^{j}、C_{4*i+17}^{j}、C_{4*i+2}^{j}、C_{4*i+18}^{j}、C_{4*i+3}^{j}和C_{4*i+19}^{j}在复用得到的第一数据流 i 中连续。即经过 8：1 符号复用后第一数据流i的数据顺序为 $C_{4*i}^{0}, C_{4*i+16}^{0}, C_{4*i+1}^{0}, C_{4*i+17}^{0}, \ldots, C_{4*i}^{j}, C_{4*i+16}^{j}, C_{4*i+1}^{j}, C_{4*i+17}^{j}, C_{4*i+2}^{j}, C_{4*i+18}^{j}, C_{4*i+3}^{j}, C_{4*i+19}^{j}, \ldots$，其中$0 \leq i \leq 3$。

需要说明的是，上述 8:1 符号复用（symbol-mux）也可以采用多级符号复用进行实现，比如采用 3 级的 2:1 符号复用。

实施方式 12：在实施方式 11 基础上，本实施方式 12 提供了分组交织的另一种具体实现方式。

从第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流、第 4*i+16 条通道数据流、第 4*i+17 条通道数据流、第 4*i+18 条通道数据流、第 4*i+19 条通道数据流（0≤i＜3）各获取 a=2 个连续的符号共 16 个符号，然后交织为第一数据流 i 中连续的 16 个符号。在一些具体应用场景下，通道数据对齐处理是基于 2 个 RS 符号对齐处理。此时，上述第 4*i 条通道数据流所获取的 2 个连续符号中的第 0 个符号、第 4*i+1 条通道数据流所获取的 2 个连续符号中的第 0 个符号来自不同 RS 码字，第 4*i 条通道数据流所获取的 2 个连续符号中的第 1 个符号、第 4*i+1 条通道数据流所获取的 2 个连续符号中的第 1 个符号来自不同 RS 码字；第 4*i+2 条通道数据流所获取的 2 个连续符号中的第 0 个符号、第 4*i+3 条通道数据流所获取的 2 个连续符号中的第 0 个符号来自不同 RS 码字，第 4*i+2 条通道数据流所获取的 2 个连续符号中的第 1 个符号、第 4*i+3 条通道数据流所获取的 2 个连

续符号中的第 1 个符号来自不同 RS 码字；上述第 4*i+16 条通道数据流所获取的 2 个连续符号中的第 0 个符号、第 4*i+17 条通道数据流所获取的 2 个连续符号中的第 0 个符号来自不同 RS 码字，第 4*i+16 条通道数据流所获取的 2 个连续符号中的第 1 个符号、第 4*i+17 条通道数据流所获取的 2 个连续符号中的第 1 个符号来自不同 RS 码字；第 4*i+18 条通道数据流所获取的 2 个连续符号中的第 0 个符号、第 4*i+19 条通道数据流所获取的 2 个连续符号中的第 0 个符号来自不同 RS 码字，第 4*i+18 条通道数据流所获取的 2 个连续符号中的第 1 个符号、第 4*i+19 条通道数据流所获取的 2 个连续符号中的第 1 个符号来自不同 RS 码字。

图 69 为本申请实施例中分组交织的再一种应用场景示意图。结合上述特征，分组交织的另一种具体实施方式如图 69 中的(a)示例所示，上述的第一数据流 i 中连续的 16 个符号中，第 0 和第 8 个符号来自第 4*i 条通道数据流，第 1 和第 9 个符号来自第 2*i+16 条通道数据流，第 2 和第 10 个符号来自第 4*i+1 条通道数据流，第 3 和第 11 个符号来自第 4*i+17 条通道数据流，第 4 和第 12 个符号来自第 4*i+2 条通道数据流，第 5 和第 13 个符号来自第 4*i+18 条通道数据流，第 6 和第 14 个符号来自第 4*i+3 条通道数据流，第 7 和第 15 个符号来自第 4*i+19 条通道数据流。需要说明的是，上述实现方式，等价于，从所述 8 条通道数据流中分别获取 1 个符号，进行交织得到数据流 i 中的连续 8 个符号，其中所述的第一数据流 i 中连续的 8 个符号中，第 0 个符号来自第 4*i 条通道数据流，第 1 个符号来自第 4*i+16 条通道数据流，第 2 个符号来自第 4*i+1 条通道数据流，第 3 个符号来自第 4*i+17 条通道数据流，第 4 个符号来自第 4*i+2 条通道数据流，第 5 个符号来自第 4*i+18 条通道数据流，第 6 个符号来自第 4*i+3 条通道数据流，第 7 个符号来自第 4*i+19 条通道数据流。

分组交织的再一种具体实施方式如图 69 中的（b）示例所示，上述的第一数据流 i 中连续的 16 个符号中，第 0 和第 8 个符号来自第 4*i 条通道数据流，第 1 和第 9 个符号来自第 4*i+1 条通道数据流，第 2 和第 10 个符号来自第 4*i+16 条通道数据流，第 3 和第 11 个符号来自第 4*i+17 条通道数据流，第 4 和第 12 个符号来自第 4*i+2 条通道数据流，第 5 和第 13 个符号来自第 4*i+3 条通道数据流，第 6 和第 14 个符号来自第 4*i+18 条通道数据流，第 7 和第 15 个符号来自第 4*i+19 条通道数据流。需要说明的是，上述实现方式，等价于，从所述 8 条通道数据流中分别获取 1 个符号，进行交织得到数据流 i 中的连续 8 个符号，其中所述的第一数据流 i 中连续的 8 个符号中，第 0 个符号来自第 4*i 条通道数据流，第 1 个符号来自第 4*i+1 条通道数据流，第 2 个符号来自第 4*i+16 条通道数据流，第 3 个符号来自第 4*i+17 条通道数据流，第 4 个符号来自第 4*i+2 条通道数据流，第 5 个符号来自第 4*i+3 条通道数据流，第 6 个符号来自第 4*i+18 条通道数据流，第 7 个符号来自第 4*i+19 条通道数据流。

需要说明的是，所述分组交织也可采用符号复用（symbol-mux）实现。符号复用从 8 条通道数据流中分别获取 1 个符号，其中C_{4*i}^{j}表示第$4*i$条通道数据流中第j个符号，C_{4*i+16}^{j}表示第$4*i$+16 条通道数据流中第j个符号，C_{4*i+1}^{j}表示第$4*i+1$条通道数据流中第j个符号，C_{4*i+17}^{j}表示第$4*i+17$条通道数据流中第j个符号，C_{4*i+2}^{j}表示第$4*i+2$条通道数据流中第j个符号，C_{4*i+18}^{j}表示第$4*i$+18 条通道数据流中第j个符号，C_{4*i+3}^{j}表示第$4*i+3$条通道数据流中第j个符号，C_{4*i+19}^{j}表示第$4*i+19$条通道数据流中第j个符号，$j\geq 0$，且C_{4*i}^{j}、C_{4*i+16}^{j}、C_{4*i+1}^{j}、C_{4*i+17}^{j}、

C^{j}_{4*i+2}、C^{j}_{4*i+18}、C^{j}_{4*i+3}、C^{j}_{4*i+19}在 8:1 符号复用得到的第一数据流中连续。即经过 8：1 符号复用后第一数据流i的数据顺序为

$C^{0}_{4*i}, C^{0}_{4*i+16}, C^{0}_{4*i+1}, C^{0}_{4*i+17}, C^{0}_{4*i+2}, C^{0}_{4*i+18}, C^{0}_{4*i+3}, C^{0}_{4*i+19}, \ldots, C^{j}_{4*i}, C^{j}_{4*i+16}, C^{j}_{4*i+1},$

$C^{j}_{4*i+17}, C^{j}_{4*i+2}, C^{j}_{4*i+18}, C^{j}_{4*i+3}, C^{j}_{4*i+19}, \ldots,$，其中$0 \le i \le 3$。其中$C^{j}_{4*i}$、$C^{j}_{4*i+16}$、$C^{j}_{4*i+1}$和$C^{j}_{4*i+17}$来自 4 个不同的 RS 码字，$C^{j}_{4*i+2}$、$C^{j}_{4*i+18}$、$C^{j}_{4*i+3}$和$C^{j}_{4*i+19}$来自 4 个不同的 RS 码字。另一种具体的符号复用是符号复用从 8 条通道数据流中分别获取 1 个符号，C^{j}_{4*i}、C^{j}_{4*i+1}、C^{j}_{4*i+16}和C^{j}_{4*i+17}、C^{j}_{4*i+2}、C^{j}_{4*i+3}、C^{j}_{4*i+18}、C^{j}_{4*i+19}在在 8:1 符号复用得到的第一数据流中连续。即经过 8：1 符号复用后第一数据流i的数据顺序为

$C^{0}_{4*i}, C^{0}_{4*i+1}, C^{0}_{4*i+16}, C^{0}_{4*i+17}, C^{0}_{4*i+2}, C^{0}_{4*i+3}, C^{0}_{4*i+18}, C^{0}_{4*i+19}, \ldots, C^{j}_{4*i}, C^{j}_{4*i+1}, C^{j}_{4*i+16},$

$C^{j}_{4*i+17}, C^{j}_{4*i+2}, C^{j}_{4*i+3}, C^{j}_{4*i+18}, C^{j}_{4*i+19}, \ldots,$ 其中$0 \le i \le 3$。

实施方式 13：以 n=32 为例，32 条通道数据流中位于奇数通道的 16 条通道数据流来自相同的码字，32 条通道数据流中位于偶数通道的 16 条通道数据流来自相同的码字，位于奇数通道的数据流和位于偶数通道的数据流来自不同的码字。即通道数据流 2*i 共 16 条通道数据流来自相同的码字，通道数据流 2*i+1 共 16 条通道数据流来自相同的码字，$0 \le i < 16$。

图 70 为本申请实施例中分组交织的再一种应用场景示意图。如图 70 所示的 1 个分组交织器用于对 8 条通道数据流进行分组交织得到 1 条第一数据流，分组交织共需要用到 4 个如图 70 所示的分组交织器。其中，所述 8 条通道数据流中，4 条通道数据流来自通道数据流 0-通道数据流 31 中的 4 条奇数通道数据流，另 4 条通道数据流来自通道数据流 0-通道数据流 31 中的 4 条偶数通道数据流。分组交织器从所述 8 条通道数据流各取 a=2 个连续的符号共 16 个符号，然后映射为所述第一数据流中连续的 D=160 个比特，即 d=16 个符号。其中，分组交织得到的 1 条第一数据流中每连续的 d=16 个符号来自至少 v=4 个不同的码字，且每连续的 L2=544 个符号来自至少 v=4 的不同的码字。进一步有分组交织得到的 1 条第一数据流中上述每连续的 d=16 个符号中的每连续的 4 个符号来自 4 个不同的 RS 码字,即每连续 16 个符号中的第 0 个符号到第 3 个符号分别来自不同的 RS 码字，第 4 个符号到第 7 个符号分别来自不同的 RS 码字，第 8 个符号到第 11 个符号分别来自不同的 RS 码字，第 12 个符号到第 15 个符号分别来自不同的 RS 码字。一种具体实现方式是，分组交织器 i 对第 8*i 条通道数据流、第 8*i+1 条通道数据流、第 8*i+2 条通道数据流、第 8*i+3 条通道数据流、第 8*i+4 条通道数据流、第 8*i+5 条通道数据流、第 8*i+6 条通道数据流、第 8*i+7 条通道数据流进行分组交织得到 1 条第一数据流，总共得到 m=4 条第一数据流， 0≤i≤3。

分组交织的一种具体交织方式为：从第 8*i 条通道数据流、第 8*i+1 条通道数据流、第 8*i+2 条通道数据流、第 8*i+3 条通道数据流、第 8*i+4 条通道数据流、第 8*i+5 条通道数据流、第 8*i+6 条通道数据流、第 8*i+7 条通道数据流（0≤i＜3）各获取 a=2 个连续的符号共 16 个符号，然后交织为第一数据流 i 中连续的 16 个符号。一种具体实施方式为，第一数据流 i 中连续的 16 个符号中，第 0 和第 1 个符号来自第 8*i 条通道数据流，第 2 和第 3 个符号来自第 8*i+1 条通道数据流，第 4 和第 5 个符号来自第 8*i+2 条通道数据流，第 6 和第 7

个符号来自第 8*i+3 条通道数据流，第 8 和第 9 个符号来自第 8*i+4 条通道数据流，第 10 和第 11 个符号来自第 8*i+5 条通道数据流，第 12 和第 13 个符号来自第 8*i+6 条通道数据流，第 14 和第 15 个符号来自第 8*i+7 条通道数据流。

图 71 为本申请实施例中分组交织的再一种应用场景示意图。需要说明的是，所述分组交织也可采用如图 71 所示的符号复用（symbol-mux）实现，其中C_{8*i}^{j}表示第$8*i$条通道数据流中第j组连续的 2 个符号，C_{8*i+1}^{j}表示第$8*i$+1 条通道数据流中第j组连续的 2 个符号，C_{8*i+2}^{j}表示第$8*i+2$条通道数据流中第j组连续的 2 个符号，C_{8*i+3}^{j}表示第$8*i+3$条通道数据流中第j组连续的 2 个符号，C_{8*i+4}^{j}表示第$8*i$+4 条通道数据流中第j组连续的 2 个符号，C_{8*i+5}^{j}表示第$8*i+5$条通道数据流中第j组连续的 2 个符号，C_{8*i+6}^{j}表示第$8*i+6$条通道数据流中第j组连续的2个符号，C_{8*i+7}^{j}表示第$8*i+7$条通道数据流中第j组连续的2个符号，$j\geq 0$，且C_{8*i}^{j}、C_{8*i+1}^{j}、C_{8*i+2}^{j}、C_{8*i+3}^{j}、C_{8*i+4}^{j}、C_{8*i+5}^{j}、C_{8*i+6}^{j}和C_{8*i+7}^{j}在复用得到的第一数据流 i 中连续。即经过 8：1 符号复用后第一数据流i的数据顺序为

$C_{8*i}^{0}, C_{8*i+1}^{0}, C_{8*i+2}^{0}, C_{8*i+3}^{0}, C_{8*i+4}^{0}, C_{8*i+5}^{0}, C_{8*i+6}^{0}, C_{8*i+7}^{0}, \ldots, C_{8*i}^{j},$

$C_{8*i+1}^{j}, C_{8*i+2}^{j}, C_{8*i+3}^{j}, C_{8*i+4}^{j}, C_{8*i+5}^{j}, C_{8*i+6}^{j}, C_{8*i+7}^{j}, \ldots$，其中$0 \leq i \leq 3$。

需要说明的是，上述 8:1 符号复用（symbol-mux）也可以采用多级符号复用进行实现，比如采用 3 级的 2:1 符号复用。

实施方式 14：在实施方式 13 基础上，本实施方式 14 提供了分组交织的另一种具体实现方式。

从第 8*i 条通道数据流、第 8*i+1 条通道数据流、第 8*i+2 条通道数据流、第 8*i+3 条通道数据流、第 8*i+4 条通道数据流、第 8*i+5 条通道数据流、第 8*i+6 条通道数据流、第 8*i+7 条通道数据流（0≤i＜3）各获取 a=2 个连续的符号共 16 个符号，然后交织为第一数据流 i 中连续的 16 个符号。在一些具体应用场景下，通道数据对齐处理是基于 2 个 RS 符号对齐处理。此时，上述第 8*i 条通道数据流所获取的 2 个连续符号中的第 0 个符号与第 8*i+2 条通道数据流所获取的 2 个连续符号中的第 0 个符号来自不同 RS 码字，第 8*i 条通道数据流所获取的 2 个连续符号中的第 1 个符号、第 8*i+2 条通道数据流所获取的 2 个连续符号中的第 1 个符号来自不同 RS 码字；第 8*i+1 条通道数据流所获取的 2 个连续符号中的第 0 个符号与第 8*i+3 条通道数据流所获取的 2 个连续符号中的第 0 个符号来自不同 RS 码字，第 8*i+1 条通道数据流所获取的 2 个连续符号中的第 1 个符号、第 8*i+3 条通道数据流所获取的 2 个连续符号中的第 1 个符号来自不同 RS 码字；第 8*i+4 条通道数据流所获取的 2 个连续符号中的第 0 个符号与第 8*i+6 条通道数据流所获取的 2 个连续符号中的第 0 个符号来自不同 RS 码字，第 8*i+4 条通道数据流所获取的 2 个连续符号中的第 1 个符号、第 8*i+6 条通道数据流所获取的 2 个连续符号中的第 1 个符号来自不同 RS 码字；第 8*i+5 条通道数据流所获取的 2 个连续符号中的第 0 个符号与第 8*i+7 条通道数据流所获取的 2 个连续符号中的第 0 个符号来自不同 RS 码字，第 8*i+5 条通道数据流所获取的 2 个连续符号中的第 1 个符号、第 8*i+7 条通道数据流所获取的 2 个连续符号中的第 1 个符号来自不同 RS 码字。

图 72 为本申请实施例中分组交织的再一种应用场景示意图。结合上述特征，分组交织的另一种具体实施方式如图 72 中的(a)示例所示，上述的第一数据流 i 中连续的 16 个符号中，第 0 和第 8 个符号来自第 8*i 条通道数据流，第 1 和第 9 个符号来自第 8*i+1 条通道数据流，第 2 和第 10 个符号来自第 8*i+2 条通道数据流，第 3 和第 11 个符号来自第 8*i+3 条通道数据流，第 4 和第 12 个符号来自第 8*i+4 条通道数据流，第 5 和第 13 个符号来自第 8*i+5 条通道数据流，第 6 和第 14 个符号来自第 8*i+6 条通道数据流，第 7 和第 15 个符号来自第 8*i+7 条通道数据流。需要说明的是，上述实现方式，等价于，从所述 8 条通道数据流中分别获取 1 个符号，进行交织得到数据流 i 中的连续 8 个符号，其中所述的第一数据流 i 中连续的 8 个符号中，第 0 个符号来自第 8*i 条通道数据流，第 1 个符号来自第 8*i +1 条通道数据流，第 2 个符号来自第 8*i +2 条通道数据流，第 3 个符号来自第 8*i +3 条通道数据流，第 4 个符号来自第 8*i +4 条通道数据流，第 5 个符号来自第 8*i +5 条通道数据流，第 6 个符号来自第 8*i +6 条通道数据流，第 7 个符号来自第 8*i +7 条通道数据流。

分组交织的再一种具体实施方式如图 72 中的（b）示例所示，上述的第一数据流 i 中连续的 16 个符号中，第 0 和第 8 个符号来自第 8*i 条通道数据流，第 1 和第 9 个符号来自第 8*i+2 条通道数据流，第 2 和第 10 个符号来自第 8*i+1 条通道数据流，第 3 和第 11 个符号来自第 8*i+3 条通道数据流，第 4 和第 12 个符号来自第 8*i+4 条通道数据流，第 5 和第 13 个符号来自第 8*i+6 条通道数据流，第 6 和第 14 个符号来自第 8*i+5 条通道数据流，第 7 和第 15 个符号来自第 8*i+7 条通道数据流。需要说明的是，上述实现方式，等价于，从所述 8 条通道数据流中分别获取 1 个符号，进行交织得到数据流 i 中的连续 8 个符号，其中所述的第一数据流 i 中连续的 8 个符号中，第 0 个符号来自第 8*i 条通道数据流，第 1 个符号来自第 8*i +2 条通道数据流，第 2 个符号来自第 8*i +1 条通道数据流，第 3 个符号来自第 8*i +3 条通道数据流，第 4 个符号来自第 8*i +4 条通道数据流，第 5 个符号来自第 8*i +6 条通道数据流，第 6 个符号来自第 8*i +5 条通道数据流，第 7 个符号来自第 8*i +7 条通道数据流。

需要说明的是，所述分组交织也可采用符号复用（symbol-mux）实现，其中C_{8*i}^{j}表示第$8*i$条通道数据流中第j个符号，C_{8*i+1}^{j}表示第$8*i+1$条通道数据流中第j个符号，C_{8*i+2}^{j}表示第$8*i+2$条通道数据流中第j个符号，C_{8*i+3}^{j}表示第$8*i+3$条通道数据流中第j个符号，C_{8*i+4}^{j}表示第$8*i+4$条通道数据流中第j个符号，C_{8*i+5}^{j}表示第$8*i+5$条通道数据流中第j个符号，C_{8*i+6}^{j}表示第$8*i+6$条通道数据流中第j个符号，C_{8*i+7}^{j}表示第$8*i+7$条通道数据流中第j个符号，$j\geq 0$，且C_{8*i}^{j}、C_{8*i+1}^{j}、C_{8*i+2}^{j}、C_{8*i+3}^{j}、C_{8*i+4}^{j}、C_{8*i+5}^{j}、C_{8*i+6}^{j}和C_{8*i+7}^{j}在复用得到的第一数据流 i 中连续。即经过 8：1 符号复用后第一数据流i的数据顺序为

$C_{8*i}^{0}, C_{8*i+1}^{0}, C_{8*i+2}^{0}, C_{8*i+3}^{0}, C_{8*i+4}^{0}, C_{8*i+5}^{0}, C_{8*i+6}^{0}, C_{8*i+7}^{0}, \ldots, C_{8*i}^{j}$,

$C_{8*i+1}^{j}, C_{8*i+2}^{j}, C_{8*i+3}^{j}, C_{8*i+4}^{j}, C_{8*i+5}^{j}, C_{8*i+6}^{j}, C_{8*i+7}^{j}, \ldots$，其中$0 \leq i \leq 3$。另一种具体的符号复用是符号复用从 8 条通道数据流中分别获取 1 个符号，C_{8*i}^{j}、C_{8*i+2}^{j}、C_{8*i+1}^{j}、C_{8*i+3}^{j}、C_{8*i+4}^{j}、C_{8*i+6}^{j}、C_{8*i+5}^{j}和C_{8*i+7}^{j}在 8:1 符号复用得到的第一数据流中连续。即经过 8：1 符号复用后第一数据流i的数据顺序为

$C_{4*i}^{0}, C_{4*i+1}^{0}, C_{4*i+16}^{0}, C_{4*i+17}^{0}, C_{4*i+2}^{0}, C_{4*i+3}^{0}, C_{4*i+18}^{0}, C_{4*i+19}^{0}, \ldots, C_{4*i}^{j}, C_{4*i+1}^{j}, C_{4*i+16}^{j}$,

$C_{4*i+17}^{j}, C_{4*i+2}^{j}, C_{4*i+3}^{j}, C_{4*i+18}^{j}, C_{4*i+19}^{j}, \ldots$，其中$0 \le i \le 3$。

需要说明的是，在一些具体实施场景下，发送设备 01 发送的 1*800GE 业务的时候，32 个 PCS 通道数据流在进行 PMA4:1 复用后通过通道连接单元接口 AUI 送到发端处理模块。在进行上述 PMA4：1 复用时，4 条输入数据流中 2 条来自第 0-15 条 PCS 通道数据流中的 2 条，另 2 条来自第 16-31 条 PCS 通道数据流中的 2 条。在发端处理模块的数据处理时，可结合上述“PMA4：1 复用时，4 条输入数据流中 2 条来自第 0-15 条 PCS 通道数据流中的 2 条，另 2 条来自第 16-31 条 PCS 通道数据流中的 2 条”特征，先进行 de-mux 后恢复出 32 个 PCS 通道数据流，其中每条物理通道数据流解复用 de-mux 得到 4 条 PCS 通道数据流，然后利用 PCS 通道已知的对齐标识（Alignment marker）进行通道数据流的标识锁定（alignment lock）；接着不需进行通道重排序，直接将上述两条物理通道数据流解复用 de-mux 得到的 8 条 PCS 通道数据流进行实施方式 11 或者实施列 12 或者实施列 13 或者实施列 14 所述的分组交织（或符号复用），将 8 条通道数据流进行分组交织（或符号复用）得到 1 条第一数据流，使得所述第一数据流中连续的 16 个符号来自至少 4 个 RS 码字；其具体实现方式可参照图 67 或 68 或 69 或 70 或者 71 或者 72，其具体实现方式本领域普通技术人员可知，此处不再赘述。

5102、对 m 条第一数据流分别进行卷积交织得到 m 条第二数据流。

图 73 为本申请实施例中对 m 条第一数据流分别进行卷积交织的一种结构示意图。如图 73 所示，可以通过 m 个卷积交织器分别对 m 条第一数据流进行卷积交织。每个卷积交织器分别对输入的第一数据流以 d 个符号为单位进行卷积交织处理，得到数据顺序打乱的第二数据流，其中 d 个符号为分组交织器进行一次分组交织交织操作得到 d 个符号，d=D/M。

需要说明的是，在本实施例中，每个卷积交织器都采用类似的方式对输入的第一数据流进行卷积交织。具体地，每个卷积交织器包括 p 条延迟线（delay line），每个卷积交织器根据 p 条延迟线对输入的第一数据流进行延迟得到第二数据流。其中，p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q。每个存储单元用于存储 d 个符号。每条通道数据流中的符号按照 p 条延迟线的序号依次输入到 p 条延迟线，每条延迟线单次输入 d 个符号且单次输出 d 个符号，1 条所述第二数据流中连续的 p*d 个符号包括每条延迟线输出的 d 个符号。其中，Q 为大于或等于 1 的整数。例如，p 条延迟线分别包括 0 个存储单元、Q 个存储单元、2Q 个存储单元、…、(p-1)Q 个存储单元，每个存储单元用于存储 d 个符号，那么，p 条延迟线分别对应 p 个延迟取值，延迟取值包括 0 个符号、Q×d 个符号、2Q×d 个符号、…、(p-1)Q×d 个符号。应理解，延迟线的延迟取值所包括的符号数量越多，就说明这条延迟线对数据流的延迟（也称为时延）越长。应理解，当延迟线没有包含存储单元时，延迟线的延迟为 0 个符号，即为无延迟透传。

需要说明的是，在同一个时刻，卷积交织器输入和输出的开关（switch）位于同一条延迟线上，待当前的延迟线单次输入 d 个符号并输出 d 符号后，再将开关切换到下一条延迟线上，以保证每条通道数据流中的符号按照 p 条延迟线的序号依次输入到 p 条延迟线，且第一数据流中连续的 p*d 个符号包括每条延迟线输出的 d 个符号。其具体数据读出写入操作如下：

从当前延迟线最靠近输出端口的存储单元读出 d 个符号。将当前延迟线中每个存储单元存储的 d 个符号转移至后一个存储单元。再将 d 个符号写入当前延迟线中最靠近输入端口的存储单元。之后，切换到下一条延迟线并重复上述操作，依次类推。

在一种可能的实施方式中，卷积交织器的结构如上述图 12（a）所示，p 条延迟线中存储单元的数量按照 p 条延迟线的序号依次递减。即延迟线 0 具有(p-1)Q 个存储单元，每一条延迟线依次减少 Q 个存储单元，延迟线 p-1 具有 0 个存储单元。在该示例中，$d(p*Q+1)\geq L2$，L2=t/s*L1，使得卷积交织器r输出的第二数据流r中连续的 d*p 个符号最多来自不同的 v*p 个码字，$0\leq r\leq m-1$。

在另一种可能的实施方式中，卷积交织器的结构如上述图 12（b）所示，p 条延迟线中存储单元的数量按照 p 条延迟线的序号依次递增。即延迟线 0 具有 0 个存储单元，每一条延迟线依次增加 Q 个存储单元，延迟线 p-1 具有(p-1)Q 个存储单元。在该示例中，$d(p*Q-1)\geq L2$，L2=t/s*L1，使得卷积交织器r输出的第二数据流r中连续的 d*p 个符号最多来自不同的 v*p 个码字，$0\leq r\leq n-1$。

应理解，当采用相同的参数 p、Q 和 d 时，图 12（a）的卷积交织处理和图 12（b）的卷积交织处理相互是对方的逆操作。也就是说，当发端处理模块采用图 12（a）所示的卷积交织结构，其收端处理模块对应的卷积解交织采用如图图 12（b）所示的结构。同理，当发端处理模块采用图 12（b）所示的卷积交织结构，其收端处理模块对应的卷积解交织采用如图 12（a）所示的结构。

还应理解，n 个卷积交织器中的任一个卷积交织器可以采用上述图 12（a）或图 12（b）中的一种。在实际应用中，n 个卷积交织器可以均采用图 12（a）所示的结构；或者，n 个卷积交织器也可以均采用图 12（b）所示的结构；又或者，也可以部分的卷积交织器采用图 12（a）所示的结构，剩余部分的卷积交织器采用图 12（b）所示的结构。

为了便于介绍，下文中涉及卷积交织的实施例均以 n 个卷积交织器采用如图 12（a）所示结构为例进行说明，当然可以简单扩展到上述列举的其他结构，其具体实现方式本领域普通技术人员可知，此处不再赘述。下面介绍几种卷积交织的具体实施例。

实施例 1：实施例 1 提供的卷积交织是在上述步骤 5101 中实施方式 1 提供的分组交织的基础上实现的。

图 74 为本申请实施例中卷积交织器的一种实施例示意图。如图 74 所示，卷积交织器包含 p=3 条延迟线（delay line）。这 p=3 条延迟线分别包括 2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=4 个符号（40bits）。也就是延迟线 0 的延迟取值为 8Q 个符号，延迟线 1 的延迟取值为 4Q 个符号，延迟线 2 的延迟取值为 0 个符号即无延迟。

如图 74 所示的 $C_r()$ 表示第一数据流r（$0\leq r\leq m-1$）中一个符号（10bits）。比如，$C_r(12t)$，$C_r(12t+1)$，$C_r(12t+2)$，$C_r(12t+3)$表示第一数据流r当前输入给延迟线 0 的 4 个符号且$C_r(12t-24Q)$，$C_r(12t-24Q+1)$，$C_r(12t-24Q+2)$，$C_r(12t-24Q+3)$为延迟线 0 输出的 4 个符号；$C_r(12t+4)$，$C_r(12t+5)$，$C_r(12t+6)$，$C_r(12t+7)$表示第一数据流r接下来输入给延迟线 1 的 4 个符号且$C_r(12t-12Q+4)$，$C_r(12t-12Q+5)$，$C_r(12t-12Q+6)$，

$C_r(12t-12Q+7)$为延迟线 1 输出的 4 个符号；$C_r(12t+8)$，$C_r(12t+9)$，$C_r(12t+10)$，$C_r(12t+11)$表示第一数据流r再接下来输入给延迟线 2 的 4 个符号且$C_r(12t+8)$，$C_r(12t+9)$，$C_r(12t+10)$，$C_r(12t+11)$为延迟线 2 输出的 4 个符号；$C_r(12t+12)$，$C_r(12t+13)$，$C_r(12t+14)$，$C_r(12t+15)$表示第一数据流r再随后输入给延迟线 0 的 4 个符号且$C_r(12t-24Q+12)$，$C_r(12t-24Q+13)$，$C_r(12t-24Q+14)$，$C_r(12t-24Q+15)$为延迟线 0 输出的 4 个符号，依次类推。结合第一数据流中 RS 分布规则，则当$d（pQ+1）\geq 136$即$Q\geq 11$时，卷积交织输出的$C_r(12t-24Q)$，$C_r(12t-24Q+1)$，$C_r(12t-24Q+2)$，$C_r(12t-24Q+3)$，$C_r(12t-12Q+4)$，$C_r(12t-12Q+5)$，$C_r(12t-12Q+6)$，$C_r(12t-12Q+7)$，$C_r(12t+8)$，$C_r(12t+9)$，$C_r(12t+10)$，$C_r(12t+11)$总共 12 个符号来自 12 个不同的 RS 码字。

在一种可能的实施方式中，图 75 为本申请实施例中卷积交织器的另一种实施例示意图。如图 75 所示，在上述图 74 所示实施例的基础上选取$Q=11$，其对应的交织时延约为 22*4*3/2=132 个 RS 符号，其等效在 1*800GE 业务时交织和解交织总时延为 50ns。

实施例 2：本实施例为在实施例 1 的基础采用新设计的卷积交织器，从而得到一种更低时延但纠错性能次优的方案。

图 76 为本申请实施例中卷积交织器的另一种实施例示意图。如图 76 所示，卷积交织器包含 p=2 条延迟线（delay line）。这 p=2 条延迟线分别包括 Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=4 个符号（40bits）。也就是延迟线 0 的延迟取值为 4Q 个符号，延迟线 1 的延迟取值为 0 个符号即无延迟。

如图 76 所示的$C_r()$表示第一数据流$r（0\leq r\leq m-1）$中一个符号（10bits）。比如，$C_r(8t)$，$C_r(8t+1)$，$C_r(8t+2)$，$C_r(8t+3)$表示第一数据流 r 当前输入给延迟线 0 的 4 个符号(40bits)且$C_r(8t-8Q)$，$C_r(8t-8Q+1)$，$C_r(8t-8Q+2)$，$C_r(8t-8Q+3)$为延迟线 0 输出的 4 个符号；$C_r(8t+4)$，$C_r(8t+5)$，$C_r(8t+6)$，$C_r(8t+7)$表示第一数据流 r 中接下来输入给延迟线 1 的 4 个符号且$C_r(8t+4)$，$C_r(8t+5)$，$C_r(8t+6)$，$C_r(8t+7)$为延迟线 1 输出的个 RS 符号；$C_r(8t+8)$，$C_r(8t+9)$，$C_r(8t+10)$，$C_r(8t+11)$表示第一数据流 r 中再接下来输入给延迟线 0 的 4 个符号且$C_r(8t-8Q+8)$，$C_r(8t-8Q+9)$，$C_r(8t-8Q+10)$，$C_r(8t-8Q+11)$为延迟线 0 输出的 4 个 RS 符号；依次类推。结合第一数据流中 RS 分布规则，则当$d（pQ+1）\geq 136$即$Q\geq 17$时，卷积交织输出的$C_r(8t-8Q)$，$C_r(8t-8Q+1)$，$C_r(8t-8Q+2)$，$C_r(8t-8Q+3)$，$C_r(8t+4)$，$C_r(8t+5)$，$C_r(8t+6)$，$C_r(8t+7)$总共 8 个符号来自 8 个不同的 RS 码字。

在一种可能的实施方式中，图 77 为本申请实施例中卷积交织器的另一种实施例示意图。如图 77 所示，在上述图 76 所示实施例的基础上选取$Q=17$，其对应的交织时延约为 17*4*2/2=68 个 RS 符号，其等效在 1*800GE 业务交织和解交织总时延为约 26ns。

实施例 3：实施例 3 提供的卷积交织是在上述步骤 5101 中实施方式 2 提供的分组交织的基础上实现的。

图 78 为本申请实施例中卷积交织器的另一种实施例示意图。如图 78 所示，卷积交织器

包含 p=3 条延迟线（delay line）。这 p=3 条延迟线分别包括 2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=2 个符号。也就是，延迟线 0 的延迟取值为 4Q 个符号，延迟线 1 的延迟取值为 2Q 个符号，延迟线 2 的延迟取值为 0 个符号即无延迟。

如图 78 所示的 $S_r()$ 表示第一数据流r（$0 \le r \le m-1$）中连续的 2 个符号共 20 比特，且此 20 比特本实施例给出的分组交织模块进行一次分组交织操作得到的 20 比特。卷积交织的过程为$S_r(3t)$表示第一数据流r当前输入给延迟线 0 的 2 个符号且$S_r(3t-6Q)$为延迟线 0 输出的 2 个 RS 符号；$S_r(3t+1)$表示第一数据流r当前输入给延迟线 1 的 2 个符号且$S_r(3t-3Q+1)$为延迟线 1 输出的 2 个符号；$S_r(3t+2)$表示第一数据流r当前输入给延迟线 2 的 2 个符号且$S_r(3t+2)$为延迟线 2 输出的 2 个符号；$S_r(3t+3)$表示第一数据流r当前输入给延迟线 0 的 2 个符号且$S_r(3t-6Q+3)$为延迟线 0 输出的 2 个符号；依次类推。结合第一数据流中 RS 分布规则，则当$d（pQ+1）\ge 68$即$Q \ge 11$时，卷积交织连续输出的$S_r(3t-6Q)$，$S_r(3t-3Q+1)$，$S_r(3t+2)$，$S_r(3t-6Q+3)$，$S_r(3t-3Q+4)$，$S_r(3t+5)$总共 12 个符号来自 12 个不同的 RS 码字。

在一种可能的实施方式中，图 79 为本申请实施例中卷积交织器的另一种实施例示意图。如图 79 所示，在上述图 78 所示实施例的基础上选取$Q=11$，其对应的交织时延约为 22*2*3/2=66 个 RS 符号，其等效在 1*800GE 业务交织和解交织总时延为 50ns。

实施例 4：本实施例为在实施例 2 的基础采用新设计的卷积交织器。

图 80 为本申请实施例中卷积交织器的另一种实施例示意图。如图 80 所示，卷积交织器包含 p=6 条延迟线（delay line）。这 p=6 条延迟线分别包括 5Q 个存储单元、4Q 个存储单元、3Q 个存储单元、2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=2 个符号（20 bits）。也就是，延迟线 0 的延迟取值为 10Q 个符号，延迟线 1 的延迟取值为 8Q 个符号，延迟线 2 的延迟取值为 6Q 个符号，延迟线 3 的延迟取值为 4Q 个符号，延迟线 4 的延迟取值为 2Q 个符号，延迟线 5 的延迟取值为 0 个符号即无延迟。

如图 80 所示的 $S_r()$ 表示第一数据流r（$0 \le r \le m-1$）中连续的 2 个符号共 20 比特，且此 20 比特本实施例给出的分组交织模块进行一次分组交织操作得到的 20 比特。卷积交织的过程为$S_r(6t)$表示第一数据流r当前输入给延迟线 0 的 2 个符号且$S_r(6t-30Q)$为延迟线 0 输出的 2 个 RS 符号；$S_r(6t+1)$表示第一数据流r当前输入给延迟线 1 的 2 个符号且$S_r(6t-24Q+1)$为延迟线 1 输出的 2 个符号；$S_r(6t+2)$表示第一数据流r当前输入给延迟线 2 的 2 个符号且$S_r(6t-18Q+2)$为延迟线 2 输出的 2 个符号；$S_r(6t+3)$表示第一数据流r当前输入给延迟线 3 的 2 个符号且$S_r(6t-12Q+3)$为延迟线 3 输出的 2 个符号；$S_r(6t+4)$表示第一数据流r当前输入给延迟线 4 的 2 个符号且$S_r(6t-6Q+4)$为延迟线 4 输出的 2 个符号；$S_r(6t+5)$表示第一数据流r当前输入给延迟线 5 的 2 个符号且$S_r(6t+5)$为延迟线 5 输出的 2 个符号；$S_r(6t+6)$表示第一数据流r当前输入给延迟线 0 的 2 个符号且$S_r(6t-30Q+6)$为延迟线 0 输出的 2 个符号；依次类推。结合第一数据流中 RS 分布规则，则当$d（2pQ+1）\ge 68$即$Q \ge 3$时，卷积交织连

续输出的$S_r(6t-30Q)$，$S_r(6t-24Q+1)$，$S_r(6t-18Q+2)$，$S_r(6t-12Q+3)$，$S_r(6t-6Q+4)$，$S_r(6t+5)$总共 12 个符号来自 12 个不同的 RS 码字。

图 81 为本申请实施例中卷积交织器的另一种实施例示意图。如图 81 所示，在上述图 80 所示实施例的基础上选取$Q=3$，其对应的交织时延约为 15*2*6/2=90 个 RS 符号，其等效在 1*800GE 业务交织和解交织总时延为 67ns。

实施例 5：本实施例提供的卷积交织是在上述步骤 5101 中实施方式 1 或 3 或者 4 或者 5 提供的分组交织(或符号复用)的基础上实现的。卷积交织器包含 p=4 条延迟线(delay line)。这 p=4 条延迟线分别包括 3Q 个存储单元、2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 d=4 个符号(40bits)。结合第一数据流中 RS 分布规则，当$d(p*Q+1)\geq 136$即$Q\geq 9$时；或者$v(p*Q-1)\geq 136$即$Q\geq 9$时，卷积交织器 4 个延迟线一次轮询输出的 16 个符号来自 16 个不同的 RS 码字。

需要说明的是，上述步骤 5101 中实施方式 3、实施方式 4 和实施例方式 6 提供的分组交织可以结合步骤 5102 中实施例 1 和实施例 2 提供的卷积交织来实现。上述步骤 5101 中实施方式 5 提供的分组交织可以结合步骤 5102 中实施例 3 和实施例 4 提供的卷积交织来实现。

需要说明的是，经过分组交织后每条第一数据流中每连续的 d 个符号来自至少 v 个不同的码字，且每条第一数据流中每连续的 L2 个符号来自至少 v 个不同的码字。在一些实现场景中，v<d，如上述步骤 5101 中实施方式 7-10。一种具体实施例下，每个卷积交织器分别对输入的第一数据流以 v 个符号为单位进行卷积交织处理，得到数据顺序打乱的第二数据流。需要说明的是，所述卷积交织中每个存储单元存储 v 个符号；下面介绍 2 种卷积交织的具体实施例。

实施例 6：本实施例提供的卷积交织是在上述步骤 5101 中实施方式 7 或 8 或者 9 或者 10 提供的分组交织（或符号复用）的基础上实现的。其在实施例 1 基础上，参考图 72，结合第一数据流中 RS 分布规则，当$v(p*Q+1)\geq 272$，即$12*Q+4\geq 272$即$Q\geq 23$时，卷积交织输出的$C_r(12t-24Q)$，$C_r(12t-24Q+1)$，$C_r(12t-24Q+2)$，$C_r(12t-24Q+3)$，$C_r(12t-12Q+4)$，$C_r(12t-12Q+5)$，$C_r(12t-12Q+6)$，$C_r(12t-12Q+7)$，$C_r(12t+8)$，$C_r(12t+9)$，$C_r(12t+10)$，$C_r(12t+11)$总共 12 个符号来自 12 个不同的 RS 码字。

图 82 为本申请实施例中卷积交织器的另一种实施例示意图。在一种可能的实施方式中，在上述图 74 所示实施例的基础上选取$Q=23$，其具体卷积交织器如图 82 所示，其对应的交织时延约为 2*23*4*3/2=276 个 RS 符号，其等效在 1*800GE 业务时交织和解交织总时延约为 52ns。

实施例 7：本实施例提供的卷积交织是在上述步骤 5101 中实施方式 11 或 12 或 13 或者 14 提供的分组交织（或符号复用）的基础上实现的。其在实施例 1 基础上，参考图 74，结合第一数据流中 RS 分布规则，当$v(p*Q+1)\geq 544$，即$12*Q+4\geq 544$即$Q\geq 45$时，卷积交织输出的$C_r(12t-24Q)$，$C_r(12t-24Q+1)$，$C_r(12t-24Q+2)$，

$C_r(12t-24Q+3)$, $C_r(12t-12Q+4)$, $C_r(12t-12Q+5)$, $C_r(12t-12Q+6)$, $C_r(12t-12Q+7)$, $C_r(12t+8)$, $C_r(12t+9)$, $C_r(12t+10)$, $C_r(12t+11)$总共 12 个符号来自 12 个不同的 RS 码字。

图 83 为本申请实施例中卷积交织器的另一种实施例示意图。在一种可能的实施方式中，在上述图 74 所示实施例的基础上选取$Q=45$，其具体卷积交织器如图 83 所示，其对应的交织时延约为 2*45*4*3/2=540 个 RS 符号，其等效在 1*800GE 业务时交织和解交织总时延约为 51ns。

实施例 8：本实施例提供的卷积交织是在上述步骤 5101 中实施方式 7 或 8 或者 9 或者 10 提供的分组交织(或符号复用)的基础上实现的。卷积交织器包含 p=4 条延迟线(delay line)。这 p=4 条延迟线分别包括 3Q 个存储单元、2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 v=4 个符号(40bits)。结合第一数据流中 RS 分布规则，当$v(p*Q+1)\geq 272$即$Q\geq 17$时;或者$v(p*Q-1)\geq 272$即$Q\geq 18$时，卷积交织器 4 个延迟线一次轮询输出的 16 个符号来自 16 个不同的 RS 码字。

实施例 9：本实施例提供的卷积交织是在上述步骤 5101 中实施方式 11 或 12 或者 13 或者 14 提供的分组交织（或符号复用）的基础上实现的。卷积交织器包含 p=4 条延迟线（delay line）。这 p=4 条延迟线分别包括 3Q 个存储单元、2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 v=4 个符号（40bits）。结合第一数据流中 RS 分布规则，当$v(p*Q+1)\geq 544$即$Q\geq 34$时;或者$v(p*Q-1)\geq 272$即$Q\geq 35$时，卷积交织器 4 个延迟线一次轮询输出的 16 个符号来自 16 个不同的 RS 码字。

需要说明的是，前面的 9 个实施列每个存储单元都是存储整数个 RS 符号，当内码的码字的信息长度为RS的整数的符号的时候，且进一步内码的码字信息长度为v*p的整数倍的时候，那么在收端处理模块可以在内码同步后自动进行解卷积交织器的同步。考虑到内码信息长度不是整数个 RS 符号的时候，给出了一种便于收端处理模块进行解卷积交织同步的卷积交织方案，下面介绍 8 种卷积交织的具体实施例。

实施例 10：本实施例提供的卷积交织是在上述步骤 5101 中实施方式 1 或 3 或者 4 或者 6 提供的分组交织(或符号复用)的基础上实现的。卷积交织器包含 p=4 条延迟线(delay line)。这 p=4 条延迟线分别包括 3Q 个存储单元、2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 v=34 个比特。结合第一数据流中 RS 分布规则，当采用图 12（a）所示卷积交织器，$v(p*Q+1)\geq 1360$即$Q\geq 10$时;或者采用图 12（b）所示卷积交织器$v(p*Q-1)\geq 1360$，即$Q\geq 11$时，卷积交织器 4 个延迟线一次轮询输出的 136 比特来自 16 个不同的 RS 码字，当 Q=10 时，对应的卷积以及解卷积的时延约 75ns。

实施例 11：本实施例提供的卷积交织是在上述步骤 5101 中实施方式 1 或 3 或者 4 或者 6 提供的分组交织(或符号复用)的基础上实现的。卷积交织器包含 p=2 条延迟线(delay line)。

这 p=2 条延迟线分别包括 Q 个存储单元、0 个存储单元，每个存储单元用于存储 v=68 个比特。结合第一数据流中 RS 分布规则，结合第一数据流中 RS 分布规则，当采用图 12（a）所示卷积交织器，$v(p*Q+1)\geq 1360$即$Q\geq 10$时；或者采用图 12（b）所示卷积交织器 $v(p*Q-1)\geq 1360$，即$Q\geq 11$时，卷积交织器 2 个延迟线一次轮询输出的 136 比特至少来自 8 个不同的 RS 码字，当 Q=10 时，对应的卷积以及解卷积的时延约 26ns。

实施例 12：本实施例提供的卷积交织是在上述步骤 5101 中实施方式 2 或 5 提供的分组交织（或符号复用）的基础上实现的。卷积交织器包含 p=4 条延迟线（delay line）。这 p=4 条延迟线分别包括 3Q 个存储单元、2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 v=34 个比特。结合第一数据流中 RS 分布规则，当采用图 12（a）所示卷积交织器，$v(p*Q+1)\geq 680$即$Q\geq 5$时；或者采用图 12(b)所示卷积交织器 $v(p*Q-1)\geq 680$，即$Q\geq 6$时，卷积交织器 4 个延迟线一次轮询输出的 136 比特来自 16 个不同的 RS 码字，当 Q=5 时，对应的卷积以及解卷积的时延约 75ns。

实施例 13：本实施例提供的卷积交织是在上述步骤 5101 中实施方式 2 或 4 提供的分组交织（或符号复用）的基础上实现的。卷积交织器包含 p=2 条延迟线（delay line）。这 p=2 条延迟线分别包括 Q 个存储单元、0 个存储单元，每个存储单元用于存储 v=68 个比特。结合第一数据流中 RS 分布规则，结合第一数据流中 RS 分布规则，当采用图 12（a）所示卷积交织器，$v(p*Q+1)\geq 680$即$Q\geq 5$时；或者采用图 12(b)所示卷积交织器 $v(p*Q-1)\geq 680$，即$Q\geq 6$时，卷积交织器 2 个延迟线一次轮询输出的 136 比特至少来自 8 个不同的 RS 码字，当 Q=5 时，对应的卷积以及解卷积的时延约 26ns。

实施例 14：本实施例提供的卷积交织是在上述步骤 5101 中实施方式 7 或 8 或者 9 或者 10 提供的分组交织（或符号复用）的基础上实现的。卷积交织器包含 p=4 条延迟线（delay line）。这 p=4 条延迟线分别包括 3Q 个存储单元、2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 v=34 个比特。结合第一数据流中 RS 分布规则，当采用图 12（a）所示卷积交织器 $v(p*Q+1)\geq 2720$即$Q\geq 20$时；或者采用图 12（b）所示卷积交织器，$v(p*Q-1)\geq 2720$即$Q\geq 21$时，卷积交织器 4 个延迟线一次轮询输出的 136 比特来自 16 个不同的 RS 码字，当 Q=20 时，对应的卷积以及解卷积的时延约 75ns。

实施例 15：本实施例提供的卷积交织是在上述步骤 5101 中实施方式 7 或 8 或者 9 或者 10 提供的分组交织（或符号复用）的基础上实现的。卷积交织器包含 p=2 条延迟线（delay line）。这 p=2 条延迟线分别包括 Q 个存储单元、0 个存储单元，每个存储单元用于存储 v=68 个比特。结合第一数据流中 RS 分布规则，当 $v(p*Q+1)\geq 2720$即$Q\geq 20$时；或者 $v(p*Q-1)\geq 2720$即$Q\geq 21$时，卷积交织器 2 个延迟线一次轮询输出的 136 比特至少来自 8 个不同的 RS 码字，当 Q=40 时，对应的卷积以及解卷积的时延约 26ns。

实施例 16：本实施例提供的卷积交织是在上述步骤 5101 中实施方式 11 或 12 或者 13 或者 14 提供的分组交织（或符号复用）的基础上实现的。卷积交织器包含 p=4 条延迟线（delay line）。这 p=4 条延迟线分别包括 3Q 个存储单元、2Q 个存储单元、Q 个存储单元、0 个存储单元，每个存储单元用于存储 v=34 个比特。结合第一数据流中 RS 分布规则，当采用图 12(a)所示卷积交织器 v(p*Q+1)≥ 5440即Q ≥ 40时；或者采用图 12(b)所示卷积交织器 v(p*Q-1)≥ 5440即Q ≥ 41时，卷积交织器 4 个延迟线一次轮询输出的 136 比特来自 16 个不同的 RS 码字，当 Q=40 时，对应的卷积以及解卷积的时延约 75ns。

实施例 17：本实施例提供的卷积交织是在上述步骤 5101 中实施方式 11 或 12 或者 13 或者 14 提供的分组交织（或符号复用）的基础上实现的。卷积交织器包含 p=2 条延迟线（delay line）。这 p=4 条延迟线分别包括 Q 个存储单元、0 个存储单元，每个存储单元用于存储 v=68 个比特。结合第一数据流中 RS 分布规则，当采用图 12(a)所示卷积交织器，v(p*Q+1)≥ 5440即Q ≥ 40时；或者采用图 12（b）所示卷积交织器，v(p*Q-1)≥ 5440即Q ≥ 41时，卷积交织器 2 个延迟线一次轮询输出的 136 比特至少来自 8 个不同的 RS 码字，当 Q=40 时，对应的卷积以及解卷积的时延约 26ns。

5103、对 m 条第二数据流分别进行第二 FEC 编码得到 m 条编码数据流。

第二 FEC 编码的实施方式具体可以如图 15 所示，对 m 条第二数据流分别进行第二 FEC 编码，即上文提到的内码编码。每条第二数据流中每连续的 K 个符号映射为一个内码码字的信息数据，经过内码编码添加冗余数据，得到编码数据流，其中每个内码长度为 K 个符号的信息数据最多来自 K 个不同的 RS 码字，其中 K≥ pd，且 K 个符号为图 12（a）或者图 12（b）所示卷积交织器的 p 个延迟线轮询 K/pd 次输出的 K 个符号。编码数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括信道交织（channel interleaving）和调制映射（mapping），以提升系统抗突发错误的能力。

在一些实现场景中，v<d，如上述步骤 5101 中实施方式 7-10。一种具体实施例下，每条第二数据流中每连续的 K 个符号映射为一个内码码字的信息数据，经过内码编码添加冗余数据，得到编码数据流，其中每个内码长度为 K 个符号的信息数据最多来自 K 个不同的 RS 码字，其中 K≥ pv，且 K 个符号为卷积交织器的 p 个延迟线轮询 K/pv 次输出的 K 个符号。

对上述 16 条或者 8 条或者 4 条第二数据流分别进行内码编码，其中内码编码信息位长度为 120 比特。具体地，内码编码器分别将第二数据流中连续的 12 个符号总共 120 比特添加冗余，得到内码码字数据流。以上述图 74 和图 75 所示实施例为例，此 12 个符号为卷积交织器的 3 条延迟线一次轮询输出的 12 个连续符号，采用此种方式可以便于在如图 3（h）所示的收端处理模块在完成内码同步后进行解卷积交织的同步。在一种可能的实施方式中，采用 Hamming(128,120)进行内码编码，分别将每条第二数据流中连续的 12 个符号总共 120 比特添加 8 比特冗余得到 128 比特的码字。

在上述步骤5102中任意实施方式提供的卷积交织的基础上对上述m条第二数据流分别进行内码编码，其中内码编码信息位长度为136比特。具体地，内码编码器分别将第二数据流中连续的136比特添加冗余，得到内码码字数据流。在一种可能的实施方式中，采用Hamming(144,136)进行内码编码，分别将每条第二数据流中连续的136比特添加8比特冗余得到144比特的码字。

在上述步骤5102中提供的实施方式10到17的卷积交织的基础上对上述m条第二数据流分别进行内码编码，其中内码编码信息位长度为136比特。具体地，将卷积交织器p=4或者2条延迟线轮询一次输出的136比特数据，作为一个内码的信息数据，然后添加8比特的校验数据，得到一个长度为144比特内码码字，这样可以在收端处理模块完成内码同步后自动完成解卷积交织器的同步。

图84为本申请实施例中内码编码的一种实现方式示意图。如图84所示，在一种可能的实施方式中，采用Hamming(144,136)进行内码编码，其由伽罗华域GF(2^8)生成的hamming（255，247）缩短111比特得到。另外一种可以的实现如图84所示，待编码的136比特数据表示为B[135:0],将其每连续的2比特进行比特异或得到1比特数据C[i],总共得到68比特数据,表示为C[67:0] 其中C[i]= B[2*i]^B[2*i+1],$0 \le i \le 67$； 然后将C[67:0]作为信息数据进行hamming（76,68）编码得到8比特校验数据，表示为P[7:0]，最后将B[135:0]与P[7:0]共144比特拼接为内码编码的输出，表示为O[143:0]。其中O[135:0]来自B[135:0]，O[143:136]来自P[7:0]，hamming（76,68）由伽罗华域GF(2^7)生成的1比特CRC扩展的hamming（128，120）缩短52比特得到。最后将O[143:0]中的每连续2bit进行PAM4格雷映射得到72个PAM4符号，表示为S[71:0]。

经过内码编码的数据流进行数据处理后送入信道传输媒介进行传输。该数据处理可包括调制映射（mapping），信道交织（channel interleaving）等。比如内码编码数据流可通过信道交织，以提升系统抗突发错误的能力。以上述图74和图75所示实施例为例，发送设备01发送的1*800GE业务时，KP4 RS(544,514) + Hamming(128,120)级联码在AWGN下，为了达到纠后误比特率BER为1E-15所对应的纠前BER约为4.8E-3，性能逼近该级联FEC方案的最优性能，且交织器的时延仅为55ns。以上述图76和图77所示实施例为例，发送设备01发送的1*800GE业务时，KP4 RS(544,514) + Hamming(128,120)级联码在AWGN下，为了达到纠后误比特率BER为1E-15所对应的纠前BER约为3.8E-3，且交织器的时延仅为26ns。以上述图78和图79所示实施例为例，发送设备01发送的1*800GE业务时，KP4 RS(544,514) + Hamming(128,120)级联码在AWGN下，为了达到纠后误比特率BER为1E-15所对应的纠前BER约为4.8E-3，性能逼近该级联FEC方案的最优性能，且交织器的时延仅为50ns。以上述图80和图81所示实施例为例，发送设备01发送的1*800GE业务时，KP4 RS(544,514) + Hamming(128,120)级联码在AWGN下，为了达到纠后误比特率BER为1E-15所对应的纠前BER约为4.8E-3，性能逼近该级联FEC方案的最优性能，且交织器的时延仅为67ns。以上述步骤5101中实施方式7或8中的分组交织（符号复用）和步骤5102中实施例5中的卷积交织为

例，发送设备 01 发送的 1*800GE 业务时，KP4 RS(544,514) + Hamming(128,120)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 4.8E-3，性能逼近该级联 FEC 方案的最优性能，且交织器的时延仅为 52ns。以上述步骤 5101 中实施方式 9 或 10 中的分组交织（符号复用）和步骤 5102 中实施例 6 中的卷积交织为例，发送设备 01 发送的 1*800GE 业务时，KP4 RS(544,514) + Hamming(128,120)级联码在 AWGN 下，为了达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 4.8E-3，性能逼近该级联 FEC 方案的最优性能，且交织器的时延仅为 52ns。

在一种可能的实施方式中，当发送设备 01 发送的 2*400GE 业务的时候，发送设备内的 32 个 PCS 通道上数据流格式如图 6 所示，其中为 PCS 通道数据流 0-15 和 PCS 通道数据流 16-32 分别属于 2 个不同的 400GE 业务。发端处理模块 PMA 单元将从通道连接单元接口 AUI 进行 de-mux 后恢复出 32 个 PCS 通道数据流，然后利用 PCS 通道已知的对齐标识(Alignment marker)进行通道数据流的标识锁定（alignment lock）。然后根据对齐标识对 n=32 条 PCS 通道数据流进行通道重排序（lane reorder）处理，使得 n=32 条 PCS 通道的数据能够按照指定的顺序排列。一种具体排列方式通道数据流 0 到通道数据流 15 属于同一个 400GE 业务，通道数据流 16 到通道数据流 31 属于另外一个 400GE 业务。排序后的 32 个通道数据流送入采用上述步骤 5101 中实施方式 1、实施方式 2 和实施例方式 3 提供的分组交织和上述步骤 5102 中卷积交织，再采用 Hamming(128,120)内码编码。另外一种具体排列方式通道数据流 0 到通道数据流 7 以及通道数据流 16 到通道数据流 23 属于同一个 400GE 业务，通道数据流 8 到通道数据流 15 以及通道数据流 24 到通道数据流 31 属于另外一个 400GE 业务。排序后的 32 个通道数据流送入采用上述步骤 5101 中实施方式 4、实施方式 5 和实施例方式 6 提供的分组交织和上述步骤 5102 中的卷积交织，再采用 Hamming(128,120)内码编码。

在此情景下，以上述图 74 和图 75 所示实施例为例，每个内码的 12 个符号来自 6 个不同的 RS 码字，达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 3.5E-3，仍保持交织器的时延仅为 50ns。以上述图 78 和图 79 所示实施例为例，每个内码的 12 个符号来自 6 个不同的 RS 码字，达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 3.5E-3，仍保持交织器的时延仅为 50ns。以上述图 80 和图 81 所示实施例为例，每个内码的 12 个符号来自 6 个不同的 RS 码字，达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 3.4E-3，仍保持交织器的时延仅为 50ns； 或者采用本实施例中的分组交织和卷积交织，并将卷积交织中的参数$Q \geq 6$,则可以使得每个内码的 12 个符号信息数据来自 12 个不同 RS 码字，达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 4.8E-3，而交织器的时延增大为 135ns。

在另一种可能的实施方式中，当发送设备 01 发送的 4*200GE 业务的时候，发送设备内的 32 个 PCS 通道上数据流格式如图 7 所示，其中为 PCS 通道数据流 0-7、PCS 通道数据流 8-15、PCS 通道数据流 16-23PCS 通道数据流 24-31 分别属于 4 个不同的 200GE 业务。发端处理模块 PMA 单元将从通道连接单元接口 AUI 接收的数据进行 de-mux 后恢复出 32 个 PCS 通道数据流，然后利用 PCS 通道已知的对齐标识（Alignment marker）进行通道数据流的标识锁定（alignment lock）。然后根据对齐标识对 n=32 条 PCS 通道数据流进行通道重排序（lane reorder）处理，使得 n=32 条 PCS 通道的数据能够按照指定的顺序排列。一种具体排列方式

通道数据流 0-7、通道数据流 8-15、通道数据流 16-23、通道数据流 24-32 分别属于 4 个不同的 200GE 业务。排序后的 32 个通道数据流送入采用上述步骤 5101 中实施方式 1、实施方式 2 和实施方式 3 提供的分组交织和上述步骤 5102 中卷积交织，再采用 Hamming(128,120)内码编码。另外一种具体的排列方式为：通道数据流 0-3 以及通道数据流 16-19、通道数据流 4-7 以及通道数据流 20-23、通道数据流 8-11 以及通道数据流 24-28、通道数据流 12-15 以及通道数据流 28-31 分别属于 4 个不同的 200GE 业务，排序后的 32 个通道数据流送入采用上述步骤 5101 中实施方式 3、实施方式 4 和实施方式 5 提供的分组交织和上述步骤 5102 中卷积交织，再采用 Hamming(128,120)内码编码。

在此情景下，以上述图 74 和图 75 所示实施例为例，每个内码的 12 个符号来自 4 个不同的 RS 码字，达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 2.8E-3，仍保持交织器的时延仅为 50ns。以上述图 78 和图 79 所示实施例为例，每个内码的 12 个符号来自 4 个不同的 RS 码字，达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 2.8E-3，仍保持交织器的时延仅为 50ns。以上述图 80 和图 81 所示实施例为例，每个内码的 12 个符号来自 4 个不同的 RS 码字，达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 2.8E-3，仍保持交织器的时延仅为 50ns；或者采用本实施例中的分组交织和卷积交织，并将卷积交织中的参数 $Q \geq 6$,则可以使得每个内码的 12 个符号信息数据来自 6 个 RS 码字，达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 3.4E-3，而交织器的时延增大为 135ns；或者采用本实施例中的分组交织和卷积交织，并将卷积交织中的参数 $Q \geq 12$,则可以使得每个内码的 12 个符号信息数据来自 12 个 RS 码字,达到纠后误比特率 BER 为 1E-15 所对应的纠前 BER 约为 4.8E-3，而交织器的时延增大为 270ns。

本发明针对采用级联 FEC 的传输方案，设计一种包含分组交织和卷积交织的数据交织编码方法，使得整体级联 FEC 方案性能较好且时延极低。使得该级联 FEC 传输方案能够应用于较多的传输场景，尤其适用于对需要较低传输时延的传输场景，如低时延数据中心互联场景。整体级联 FEC 方案在处理 1x800GE 业务时相比现有技术可以在同等纠能力下时延更优，而对于 2*400GE 或者 4*200GE 业务的时候可以采用同样的交织方案获得次优的纠错性能以及更优的时延，使得本方案可以很好的适用与时延敏感的场景，比如数据中心内部互联网络。

下面介绍本申请实施例提供的数据处理装置。

图 85 为本申请实施例中数据处理装置的一种结构示意图。如图 85 所示，该数据处理装置包括分组交织器 501 和卷积交织器 502。分组交织器 501 用于执行上述步骤 5101 的操作。卷积交织器 502 用于执行上述步骤 5102 的操作。具体可以参照上述数据处理方法中关于分组交织操作和卷积交织操作的相关介绍，此处不再赘述。

应理解，本申请提供的装置也可以通过其他方式实现。例如，上述装置中的单元划分仅仅是一种逻辑功能划分，实际实现时可以有另外的划分方式，例如，多个单元或组件可以结合或可以集成到另一个系统。另外，本申请各个实施例中的各功能单元可以集成在一个处理单元中，也可以是各个独立的物理单元，也可以是两个或两个以上个功能单元集成在一个处理单元中。上述集成的单元既可以采用硬件的形式实现，也可以采用软件功能单元的形式实

现。

需要说明的是，除了上述实施例介绍的数据处理方法外，本申请还提供了另一种数据处理方法，其具体处理流程可以如上述图3（i）所示，在一些具体实施场景下，发送设备01发送的1*800GE业务的时候，32个PCS通道数据流经过PMA（32:n）处理将32/n条PCS通道数据流复用为一个数据流PMA通道数据流时，被复用的32/n条PCS通道数据流中一半PCS通道数据流来自PCS通道数据流0-15，另外一半PCS通道数据流来自PCS通道数据流16:32，从而使得每个PAM通道数据流的数据由4个RS码字交织而来。发端处理模块的物理媒体附加子层（Physical Medium Attachment，PMA）子层对来自AUI-n接口的数据进行处理后得到n条通道数据流。此处PMA子层仅需要对来自AUI-n接口的每个物理通道的数据进行时钟数据恢复(clock data recovery，CDR)，PAM4符号解调制等信号恢复操作处理得到一条通道数据流，不需要进行AM锁定、通道纠偏以及通道重排序等其他复杂操作。然后将分别对每个第一数据流进行卷积交织（convolutional interleaving）得到n条第一数据流，最后分别对n条第一数据流进行内码编码得到n条第二数据流。

图86为本申请实施例提供的数据处理方法的另一种流程示意图。

8601、对n条通道数据流分别进行卷积交织得到n条第一数据流。

具体地，根据p条延迟线对n条通道数据流分别进行延迟得到n条第一数据流。其中，每条通道数据流都是经过第一FEC编码的，即每条通道数据流都是经过外码编码的。应理解，本实施例采用的卷积交织方式与上述图51所示实施例采用的卷积交织方式类似，具体可以参考图51所示实施例的相关介绍。

作为一个示例，当n=8的时候，对应的卷积交织器可以采用上述步骤5102中提供的实施方式6，内码编码方案为将卷积交织器p=3延迟线轮询一次输出的120比特数据作为一个内码的信息数据，然后添加8比特的校验数据，得到一个长度为128比特内码码字；或者对应的卷积交织器可以采用上述步骤5102中提供的实施方式8，内码编码方案为将卷积交织器p=4延迟线轮询一次输出的160比特数据作为一个内码的信息数据，然后添加10比特的校验数据，得到一个长度为170比特内码码字，或者添加16比特的校验数据，得到一个长度为176比特的内码码字；或者对应的卷积交织器可以采用上述步骤5102中提供的实施方式14，15，内码编码方案为将卷积交织器p=2或者4条延迟线轮询一次输出的136比特数据作为一个内码的信息数据，然后添加8比特的校验数据得到码长为144比特的码字。

作为有一个示例，当n=4的时候，对应的卷积交织器可以采用上述步骤5102中提供的实施方式7，内码编码方案为将卷积交织器p=3延迟线轮询一次输出的120比特数据作为一个内码的信息数据，然后添加8比特的校验数据，得到一个长度为128比特内码码字；或者对应的卷积交织器可以采用上述步骤5102中提供的实施方式9，内码编码方案为将卷积交织器p=4延迟线轮询一次输出的160比特数据作为一个内码的信息数据，然后添加10比特的校验数据，得到一个长度为170比特内码码字，或者添加16比特的校验数据，得到一个长度为176比特的内码码字；或者对应的卷积交织器可以采用上述步骤5102中提供的实施方式16，17，内码编码方案为将卷积交织器p=2或者4条延迟线轮询一次输出的136比特数据作为一个内码的

信息数据，然后添加 8 比特的校验数据得到码长为 144 比特的码字。

8602、对 n 条第一数据流分别进行第二 FEC 编码得到 n 条第二数据流。

应理解，本实施例中的第二 FEC 编码即上文介绍的内码编码。采用上述卷积交织和内码编码方案都可以使得内码码字的信息数据的长度等于 p*U，其中 p 是卷积交织器中延迟线的数目，U 为卷积交织器中每个存储单元存储的比特数目。这样便于内码码字的信息数据与卷积交织器的 p 条延迟线轮询一次输出的 p*U 比特对齐，因此可以在编码数流中不额外添加同步头序列情况下，收端模块在利用内码码字完成内码自同步后可以自动完成解卷积交织器的同步，从而解决卷积交织器的同步的问题。

本申请还提供了一种与图 86 所示实施例对应的数据处理装置。

图 87 为本申请实施例中数据处理装置的另一种结构示意图。如图 87 所示，该数据处理装置包括卷积交织器 601 和编码器 602。卷积交织器 601 用于执行上述步骤 8601 的操作。编码器 602 用于执行上述步骤 8602 的操作。具体可以参照上述数据处理方法中关于分组交织操作和卷积交织操作的相关介绍，此处不再赘述。

应理解，本申请提供的装置也可以通过其他方式实现。例如，上述装置中的单元划分仅仅是一种逻辑功能划分，实际实现时可以有另外的划分方式，例如，多个单元或组件可以结合或可以集成到另一个系统。另外，本申请各个实施例中的各功能单元可以集成在一个处理单元中，也可以是各个独立的物理单元，也可以是两个或两个以上个功能单元集成在一个处理单元中。上述集成的单元既可以采用硬件的形式实现，也可以采用软件功能单元的形式实现。

图 88 为本申请实施例中数据处理装置的另一种结构示意图。如图 88 所示，数据处理装置包括处理器 201、存储器 202 和收发器 203。该处理器 201、存储器 202 和收发器 203 通过线路相互连接。其中，存储器 202 用于存储程序指令和数据。具体地，收发器 203 用于接收 n 条通道数据流。处理器 201 用于执行上述数据处理方法的操作。在一种可能的实施方式中，处理器 201 可以包括上述图 33 所示的卷积交织器 101 和复用器 201。在另一种可能的实施方式中，处理器 201 可以包括上述图 50（a）所示的交织模块 301 和编码器 302。在又一种可能的实施方式中，处理器 201 可以包括上述图 50（b）所示的交织模块 401 和编码器 402。在又一种可能的实施方式中，处理器 201 可以包括上述图 85 所示的分组交织器 501 和卷积交织器 502。在又一种可能的实施方式中，处理器 201 可以包括上述图 87 所示的卷积交织器 601 和编码器 602。

需要说明的是，上述图 88 中所示的处理器可以采用通用的中央处理器（Central Processing Unit，CPU），通用处理器、数字信号处理器（DSP）、专用集成电路（ASIC）、现场可编程门阵列（FPGA）或者其他可编程逻辑器件、晶体管逻辑器件、硬件部件或者其任意组合。上述图 88 中所示的存储器可以存储操作系统和其他应用程序。在通过软件或者固件来实现本申请实施例提供的技术方案时，用于实现本申请实施例提供的技术方案的程序代码保存在存储器中，并由处理器来执行。在一实施例中，处理器内部可以包括存储器。在另一

实施例中，处理器和存储器是两个独立的结构。

所属领域的技术人员可以清楚地了解到，为描述的方便和简洁，上述描述的系统，装置和单元的具体工作过程，可以参考前述方法实施例中的对应过程，在此不再赘述。

本领域普通技术人员可以理解实现上述实施例的全部或部分步骤可以通过硬件来完成，也可以通过程序来指令相关的硬件完成，所述的程序可以存储于一种计算机可读存储介质中，上述提到的存储介质可以是只读存储器，随机接入存储器等。上述的这些功能究竟以硬件还是软件方式来执行，取决于技术方案的特定应用和设计约束条件。专业技术人员可以对每个特定的应用来使用不同方法来实现所描述的功能，但是这种实现不应认为超出本申请的范围。

当使用软件实现时，上述实施例描述的方法步骤可以全部或部分地以计算机程序产品的形式实现。所述计算机程序产品包括一个或多个计算机指令。在计算机上加载和执行所述计算机程序指令时，全部或部分地产生按照本申请实施例所述的流程或功能。所述计算机可以是通用计算机、专用计算机、计算机网络、或者其他可编程装置。所述计算机指令可以存储在计算机可读存储介质中，或者从一个计算机可读存储介质向另一个计算机可读存储介质传输，例如，所述计算机指令可以从一个网站站点、计算机、服务器或数据中心通过有线（例如同轴电缆、光纤、数字用户线（DSL））或无线（例如红外、无线、微波等）方式向另一个网站站点、计算机、服务器或数据中心进行传输。所述计算机可读存储介质可以是计算机能够存取的任何可用介质或者是包含一个或多个可用介质集成的服务器、数据中心等数据存储设备。所述可用介质可以是磁性介质，(例如，软盘、硬盘、磁带)、光介质(例如，DVD)、或者半导体介质（例如固态硬盘Solid State Disk (SSD)）等。

权 利 要 求

1. 一种数据处理方法，其特征在于，包括：

对 n 条通道数据流分别进行卷积交织得到 n 条第一数据流，所述 n 为大于 1 的整数，所述 n 条通道数据流都经过第一前向纠错 FEC 编码，所述第一 FEC 编码后的每 a 个码字分布在 b 条所述通道数据流中，所述 a≤所述 b≤所述 n，所述 n 能被所述 b 整除，所述 a 为大于或等于 1 的整数，每条所述第一数据流中连续的 z 个符号来自 z 个不同的码字，所述 z 为大于 1 的整数；

对所述 n 条第一数据流中的每 K 条第一数据流进行复用得到 1 条第二数据流，以得到共 m 条所述第二数据流，其中，所述 n 条第一数据流包括 G 个第一数据流子集，不同第一数据流子集中的符号来自不同码字，m=n/K，所述 K 为大于 1 的整数，所述 G 为大于 1 的整数，每条所述第二数据流中连续的 y 个符号来自 y 个不同的码字，所述 y＞所述 z；

若 K≤G，则所述 K 条第一数据流分别来自 K 个第一数据流子集；

若 K＞G，则所述 K 条第一数据流包括每个第一数据流子集中的 K/G 条第一数据流。

2. 根据权利要求 1 所述的方法，其特征在于，对 1 条通道数据流进行卷积交织得到 1 条第一数据流包括：

根据 p 条延迟线对 1 条通道数据流进行延迟得到 1 条第一数据流，所述 p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 d 个符号，所述 z=p*d，每条通道数据流中的符号按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 d 个符号且单次输出 d 个符号，1 条所述第一数据流中连续的 p*d 个符号包括每条延迟线输出的 d 个符号，所述 Q 为大于或等于 1 的整数，所述 d 为大于或等于 1 的整数。

3. 根据权利要求 2 所述的方法，其特征在于，所述 p 条延迟线中序号最大的延迟线包括 0 个存储单元，其中，$d(p*Q+1) \geq a*N/b$，d≤a，所述 N 为所述码字的长度。

4. 根据权利要求 2 所述的方法，其特征在于，所述 p 条延迟线中序号最小的延迟线包括 0 个存储单元，其中，$d(p*Q-1) \geq a*N/b$，d≤a，所述 N 为所述码字的长度。

5. 根据权利要求 1 至 4 中任一项所述的方法，其特征在于，若 K≤G，则 y=K*z；若 K＞G，则 y=G*z。

6. 根据权利要求 1 至 5 中任一项所述的方法，其特征在于，每条所述第二数据流包括多个第二数据流符号子集，每个所述第二数据流符号子集包括 K 组符号，每组符号包括Δ个符号，其中，每个所述第二数据流符号子集中相邻两组符号来自不同的第一数据流子集；

若 K≤G，则Δ为 z 的约数；若 K>G，则Δ= z。

7. 根据权利要求 6 所述的方法，其特征在于，每个所述第二数据流符号子集中第 j 组符号来自参与复用的 K 条第一数据流中第 j 条第一数据流，0≤j≤K-1。

8. 根据权利要求 6 或 7 所述的方法，其特征在于，若 K＞G，参与复用的 K 条第一数据流中相邻的 2 条第一数据流来自不同的第一数据流子集。

9.根据权利要求6至8中任一项所述的方法，其特征在于，若K＞G，参与复用的每连续G条第一数据流分别来自不同的第一数据流子集。

10.根据权利要求1至9中任一项所述的方法，其特征在于，所述n=32，所述K=2、4或8。

11.根据权利要求1至10中任一项所述的方法，其特征在于，所述n=32，所述p=2、3、4、6或8，所述d=1或2。

12.根据权利要求1至11中任一项所述的方法，其特征在于，所述a=1或2，所述b=4、8、16。

13.根据权利要求1至12中任一项所述的方法，其特征在于，对n条通道数据流分别进行卷积交织得到n条第一数据流之前，所述方法还包括：

对所述n条通道数据流进行通道重排序，以使得所述n条通道数据流按照预设顺序排列。

14.根据权利要求1至13中任一项所述的方法，其特征在于，对n条通道数据流分别进行卷积交织得到n条第一数据流之前，所述方法还包括：

对所述n条通道数据流进行通道纠偏。

15.根据权利要求1至13中任一项所述的方法，其特征在于，对n条通道数据流分别进行卷积交织得到n条第一数据流之前，所述方法还包括：

对所述n条通道数据流进行对齐，以使得n条通道数据流中的符号对齐。

16.根据权利要求1至15中任一项所述的方法，其特征在于，得到共m条所述第二数据流之后，所述方法还包括：

对所述m条第二数据流分别进行第二FEC编码，所述第二FEC编码的信息位长度小于或等于y个符号。

17.一种数据处理装置，其特征在于，包括：卷积交织器和复用器；

所述卷积交织器用于：对n条通道数据流分别进行卷积交织得到n条第一数据流，所述n为大于1的整数，所述n条通道数据流都经过第一前向纠错FEC编码，所述第一FEC编码后的每a个码字分布在b条所述通道数据流中，所述a≤所述b≤所述n，所述n能被所述b整除，所述a为大于或等于1的整数，每条所述第一数据流中连续的z个符号来自z个不同的码字，所述z为大于1的整数；

所述复用器用于：对所述n条第一数据流中的每K条第一数据流进行复用得到1条第二数据流，以得到共m条所述第二数据流，其中，所述n条第一数据流包括G个第一数据流子集，不同第一数据流子集中的符号来自不同码字，m=n/K，所述K为大于1的整数，所述G为大于1的整数，每条所述第二数据流中连续的y个符号来自y个不同的码字，所述y＞所述z；

若K≤G，则所述K条第一数据流分别来自K个第一数据流子集；

若K＞G，则所述K条第一数据流包括每个第一数据流子集中的K/G条第一数据流。

18.根据权利要求17所述的数据处理装置，其特征在于，所述卷积交织器具体用于：

根据p条延迟线对1条通道数据流进行延迟得到1条第一数据流，所述p为大于1的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括0个存储单

元，每相邻两条延迟线的存储单元数量的差值为Q，每个存储单元用于存储d个符号，所述z=p*d，每条通道数据流中的符号按照所述p条延迟线的序号依次输入所述p条延迟线，每条延迟线单次输入d个符号且单次输出d个符号，1条所述第一数据流中连续的p*d个符号包括每条延迟线输出的d个符号，所述Q为大于或等于1的整数，所述d为大于或等于1的整数。

19.根据权利要求18所述的数据处理装置，其特征在于，所述p条延迟线中序号最大的延迟线包括0个存储单元，其中，$d(p*Q+1) \geq a*N/b$，d≤a，所述N为所述码字的长度。

20.根据权利要求18所述的数据处理装置，其特征在于，所述p条延迟线中序号最小的延迟线包括0个存储单元，其中，$d(p*Q-1) \geq a*N/b$，d≤a，所述N为所述码字的长度。

21.根据权利要求17至20中任一项所述的数据处理装置，其特征在于，若K≤G，则y=K*z；若K＞G，则y=G*z。

22.根据权利要求17至21中任一项所述的数据处理装置，其特征在于，每条所述第二数据流包括多个第二数据流符号子集，每个所述第二数据流符号子集包括K组符号，每组符号包括Δ个符号，其中，每个所述第二数据流符号子集中相邻两组符号来自不同的第一数据流子集；

若K≤G，则Δ为z的约数；若K>G，则Δ=z。

23.根据权利要求22所述的数据处理装置，其特征在于，每个所述第二数据流符号子集中第j组符号来自参与复用的K条第一数据流中第j条第一数据流，0≤j≤K-1。

24.根据权利要求22或23所述的数据处理装置，其特征在于，若K＞G，参与复用的K条第一数据流中相邻的2条第一数据流来自不同的第一数据流子集。

25.根据权利要求22至24中任一项所述的数据处理装置，其特征在于，若K＞G，参与复用的每连续G条第一数据流分别来自不同的第一数据流子集。

26.根据权利要求17至25中任一项所述的数据处理装置，其特征在于，所述n=32，所述K=2、4或8。

27.根据权利要求17至26中任一项所述的数据处理装置，其特征在于，所述n=32，所述p=2、3、4、6或8，所述d=1或2。

28.根据权利要求17至27中任一项所述的数据处理装置，其特征在于，所述a=1或2，所述b=4、8、16。

29.根据权利要求17至28中任一项所述的数据处理装置，其特征在于，所述数据处理装置还包括通道重排序单元，对n条通道数据流分别进行卷积交织得到n条第一数据流之前，所述通道重排序单元用于：

对所述n条通道数据流进行通道重排序，以使得所述n条通道数据流按照预设顺序排列。

30.根据权利要求17至29中任一项所述的数据处理装置，其特征在于，所述数据处理装置还包括通道数据对齐单元，对n条通道数据流分别进行卷积交织得到n条第一数据流之前，所述通道数据对齐单元用于：

对所述n条通道数据流进行通道纠偏。

31.根据权利要求 17 至 29 中任一项所述的数据处理装置，其特征在于，所述数据处理装置还包括通道数据对齐单元，对 n 条通道数据流分别进行卷积交织得到 n 条第一数据流之前，所述通道数据对齐单元用于：

对所述 n 条通道数据流进行对齐，以使得 n 条通道数据流中的符号对齐。

32.根据权利要求 17 至 31 中任一项所述的数据处理装置，其特征在于，所述数据处理装置还包括编码器，得到共 m 条所述第二数据流之后，所述编码器用于：

对所述 m 条第二数据流分别进行第二 FEC 编码，所述第二 FEC 编码的信息位长度小于或等于 y 个符号。

33.一种数据处理方法，其特征在于，包括：

对 n 条通道数据流进行交织处理得到 m 条目标数据流，所述 n 为 4 的倍数，所述 n 条通道数据流都经过第一前向纠错 FEC 编码，所述第一 FEC 编码后的每 a 个码字分布在 b 条所述通道数据流中，a≤b≤n，所述 n 能被所述 b 整除，每条所述目标数据流中连续的 F 个符号来自 F 个不同的码字，所述 F＞所述 a，每条所述目标数据流中连续的 F 个符号至少来自 K1 条不同的通道数据流，每条所述目标数据流中连续的 F 个符号来自所述 n 条通道数据流对齐的 n 个符号中最多 K2 个符号，所述 K1 和所述 K2 为所述 n 的约数，所述 K2 为所述 K1 的约数，每条所述目标数据流中连续的 F 个符号中最多有 K3 个符号来自相同的通道数据流，K3=[F/K1]，所述[F/K1]表示对 F/K1 的商进行向上取整的整数，所述 K3 个符号中任意两个符号在同一通道数据流上相隔最少 K4 个符号，所述 K4≥a*N*K2/n，所述 N 为所述码字长度。

34.根据权利要求 33 所述的方法，其特征在于，K1=n/4，K2=n/16。

35.根据权利要求 33 或 34 所述的方法，其特征在于，对 n 条通道数据流进行交织处理得到 m 条目标数据流包括：

对 n 条通道数据流分别进行卷积交织得到 n 条第一数据流，每条所述第一数据流中连续的 z 个符号来自至少 c 个不同的码字，所述 z 为大于 1 的整数，a≤c≤F，c*k2≥F，每条所述第一数据流中连续的 z 个符号中最多 k1/k2 个符号来自同一个所述码字；

对所述 n 条第一数据流中的每 K1 条第一数据流进行分组交织得到 S 条所述目标数据流，以得到共 m 条所述目标数据流，其中，所述 S 为大于或等于 1 的整数，m=S*n/K1，S≥k1/k2，所述 n 条第一数据流包括 K1 个第一数据流分组，同一第一数据流分组中每两条第一数据流的符号来自相同的码字，所述 K1 条第一数据流分别来自 K1 个第一数据流分组。

36.根据权利要求 35 所述的方法，其特征在于，对 1 条通道数据流进行卷积交织得到 1 条第一数据流包括：

根据 p 条延迟线对 1 条通道数据流进行延迟得到 1 条第一数据流，所述 p 为大于 1 的整数，p*a≥F/k2，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 d 个符号，所述 z=p*d，每条通道数据流中的符号按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 d 个符号且单次输出 d 个符号，1 条所述第一数据流中连续的 p*d 个符号包括每条延迟线输出的 d 个符号，所述 Q 为大于或等于 1 的整数，所述 d 为大于或等

于 1 的整数，d≤a。

37. 根据权利要求 36 所述的方法，其特征在于，所述 p 条延迟线中序号最大的延迟线包括 0 个存储单元，其中，$d(p*Q+1)\geq K4$。

38. 根据权利要求 36 所述的方法，其特征在于，所述 p 条延迟线中序号最小的延迟线包括 0 个存储单元，其中，$d(p*Q-1)\geq K4$。

39. 根据权利要求 36 至 38 中任一项所述的方法，其特征在于，参与分组交织的 K 条第一数据流包括第一符号矩阵，所述第一符号矩阵包括 K 行 B 列个符号，B=R*p*d，所述 R 为大于或等于 1 的整数，经过分组交织后的 S 条目标数据流包括第二符号矩阵，所述第二符号矩阵包括 S 行 F 列个符号，K*B=S*F；

所述第一符号矩阵中的符号至少来自 F 个不同码字，且所述第一符号矩阵中最多有 R*K1/K2 个符号来自相同的码字。

40. 根据权利要求 39 所述的方法，其特征在于，所述第二符号矩阵中每行的 F 个符号来自所述第一符号矩阵中至少$\lfloor F/K2\rfloor$列，且所述$\lfloor F/K2\rfloor$列中每列最多取K2个符号，所述$\lfloor F/K2\rfloor$表示对 F/K2的商进行向上取整的整数；

所述第二符号矩阵中每行的 F 个符号包括所述第一符号矩阵中每行的至少$\lfloor F/K1\rfloor$个符号，所述$\lfloor F/K1\rfloor$表示对 F/K1 的商进行向下取整的整数，且所述第二符号矩阵中每行的 F 个符号包括所述第一符号矩阵中每行的最多$\lfloor F/K1\rfloor$个符号，所述$\lfloor F/K1\rfloor$表示对 F/K 的商进行向上取整的整数。

41. 根据权利要求 39 或 40 所述的方法，其特征在于，所述第二符号矩阵的每一行中来自所述第一符号矩阵奇数列的符号分别位于所述第一符号矩阵的不同行，所述第二符号矩阵的每一行中来自所述第一符号矩阵偶数列的符号位于所述第一符号矩阵的不同行。

42. 根据权利要求 39 至 41 中任一项所述的方法，其特征在于，所述第二符号矩阵的每一行中由延迟取值相同的延迟线输出的符号来自所述第一符号矩阵的不同行。

43. 根据权利要求 39 至 42 中任一项所述的方法，其特征在于，所述第二符号矩阵的每一行的最多 K3 个符号来自所述第一符号矩阵的同一行，所述 K3 个符号中任意两个符号分别由延迟差值大于或等于 2*Q*d 的 2 条延迟线输出。

44. 根据权利要求 33 或 34 所述的方法，其特征在于，对 n 条通道数据流进行交织处理得到 m 条目标数据流包括：

对 n 条通道数据流进行第一分组交织得到 T 条第一数据流，每条所述第一数据流中连续的 C 个符号至少来自 E 个不同的码字，T=n/K1，所述 C 为所述 a 的倍数，E≥K2*a；

对所述 T 条第一数据流进行卷积交织得到 T 条第二数据流，每条所述第二数据流中连续的 H 个符号来自至少 F 个不同码字，F≥E，每条所述第二数据流中连续的 H 个符号最多 K1/K2 个符号来自相同码字；

对所述 T 条第二数据流中每条第二数据流进行第二分组交织得到 S 条目标数据流，以得到共 m 条目标数据流，m=T*S，$S\geq k1/K2$。

45. 根据权利要求 44 所述的方法，其特征在于，参与第一分组交织的 n 条通道数据流包

括第三符号矩阵，所述第三符号矩阵包括 n 行 A 列个符号，所述 A 为所述 a 的倍数，经过第一分组交织后的 T 条第一数据流包括第四符号矩阵，所述第四符号矩阵包括 T 行 C 列个符号，所述 T 为所述 n 的约数，n*A=T*C；

所述第三符号矩阵的一列中每连续 T 个符号为符号子矩阵，所述第四符号矩阵中每一列的 T 个符号分别与所述第三符号矩阵中每个符号子矩阵一一对应。

46. 根据权利要求 45 所述的方法，其特征在于，所述第三符号矩阵中的符号子矩阵按第一顺序排列，所述第三符号矩阵中每列的第 1 行到第 n 行包括按所述第一顺序排列的第 1 个符号子矩阵到第 n/T 个符号子矩阵，所述第三符号矩阵的相邻两列中前一列的第 n/T 个符号子矩阵到后一列的第 1 个符号子矩阵为按所述第一顺序排列的连续 2 个符号子矩阵，所述第四符号矩阵中第 1 列的 T 个符号来自所述第三符号矩阵中按所述第一顺序排列的第一个符号子矩阵，依次类推，直到所述第四符号矩阵中第 C 列的 T 个符号来自所述第三符号矩阵中按所述第一顺序排列的最后一个符号子矩阵；

或者，

所述第三符号矩阵中的符号子矩阵按第二顺序排列，所述第三符号矩阵中每 T 行的第 1 列到第 A 行包括按所述第二顺序排列的第 1 个符号子矩阵到第 A 个符号子矩阵，所述第三符号矩阵的相邻两个连续 T 行中前 T 行的第 A 个符号子矩阵到后 T 行的第 1 个符号子矩阵为按所述第二顺序排列的连续 2 个符号子矩阵，所述第四符号矩阵中第 1 列的 T 个符号来自所述第三符号矩阵中按所述第二顺序排列的第一个符号子矩阵，依次类推，直到所述第四符号矩阵中第 C 列的 T 个符号来自所述第三符号矩阵中按所述第二顺序排列的最后一个符号子矩阵。

47. 根据权利要求 44 至 46 中任一项所述的方法，其特征在于，对 1 条第一数据流进行卷积交织得到 1 条第二数据流包括：

根据 p 条延迟线对 1 条第一数据流进行延迟得到 1 条第二数据流，所述 p 为大于 1 的整数，p*F≥F，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 C 个符号，p*C=H，每条第一数据流中的符号按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 C 个符号且单次输出 C 个符号，1 条所述第二数据流中连续的 p*C 个符号包括每条延迟线输出的 C 个符号，所述 Q 为大于或等于 1 的整数。

48. 根据权利要求 47 所述的方法，其特征在于，所述 p 条延迟线中序号最大的延迟线包括 0 个存储单元，其中，$C(p*Q+1) \geq K1*K4$。

49. 根据权利要求 47 所述的方法，其特征在于，所述 p 条延迟线中序号最小的延迟线包括 0 个存储单元，其中，$C(p*Q-1) \geq K1*K4$。

50. 根据权利要求 47 至 49 中任一项所述的方法，其特征在于，每条所述第二数据流包括 R 个符号集合，每个所述符号集合包括 p 个符号子集，每个所述符号子集包括 C 个符号，p 个符号子集分别由所述 p 条延迟线输出，每个符号集合中的符号来自至少 F 个不同码字，每条所述目标数据流包括 F 个符号，R*p*C=S*F，所述 R 为大于或等于的整数；

所述目标数据流中 F 个符号来自至少$\lfloor F/K2*a) \rfloor$个不同的符号子集，且所述$\lceil F/(K2*a) \rceil$

个不同的符号子集中每个符号子集最多取K2＊a个符号，所述⌈F/（K2＊a)⌉表示对F/（K2＊a)的商进行向上取整的整数。

51.根据权利要求50所述的方法，其特征在于，所述目标数据流中F个符号包括来自第一符号子集中的第一组符号和来自第二符号子集的第二组符号，所述第一符号子集和所述第二符号子集属于同一个符号集合，所述第一符号子集和所述第二符号子集分别由两条相邻的延迟线输出，所述第一符号子集中的符号和所述第二符号子集中的符号分别按次序排列，所述第一组符号在所述第一符号子集中的次序与所述第二组符号在所述第二符号子集中的次序不同；

所述目标数据流中F个符号包括来自第三符号子集中的第三组符号和来自第四符号子集的第四组符号，所述第三符号子集和所述第四符号子集属于不同的符号集合，所述第三符号子集和所述第四符号子集由同一条延迟线输出，所述第三符号子集中的符号和所述第四符号子集中的符号分别按次序排列，所述第三组符号在所述第三符号子集中的次序与所述第四组符号在所述第四符号子集中的次序不同。

52.根据权利要求50或51所述的方法，其特征在于，每条所述目标数据流中F个符号中最多有⌈F/R⌉个符号来自相同的符号集合，所述⌈F/R⌉表示对F/R的商进行向上取整的整数。

53.根据权利要求33至52中任一项所述的方法，其特征在于，得到共m条所述目标数据流之后，所述方法还包括：

对所述m条目标数据流分别进行第二FEC编码，所述第二FEC编码的信息位长度等于F个符号。

54.一种数据处理装置，其特征在于，包括：交织模块；

所述交织模块用于：对n条通道数据流进行交织处理得到m条目标数据流，所述n为4的倍数，所述n条通道数据流都经过第一前向纠错FEC编码，所述第一FEC编码后的每a个码字分布在b条所述通道数据流中，a≤b≤n，所述n能被所述b整除，每条所述目标数据流中连续的F个符号来自F个不同的码字，所述F＞所述a，每条所述目标数据流中连续的F个符号至少来自K1条不同的通道数据流，每条所述目标数据流中连续的F个符号来自所述n条通道数据流对齐的n个符号中最多K2个符号，所述K1和所述K2为所述n的约数，所述K2为所述K1的约数，每条所述目标数据流中连续的F个符号中最多有K3个符号来自相同的通道数据流，K3=⌈F/K1⌉，所述⌈F/K1⌉表示对F/K1的商进行向上取整的整数，所述K3个符号中任意两个符号在同一通道数据流上相隔最少K4个符号，所述K4≥a*N*K2/n，N为所述码字长度。

55.根据权利要求54所述的数据处理装置，其特征在于，K1=n/4，K2=n/16。

56.根据权利要求54或55所述的数据处理装置，其特征在于，所述交织模块包括卷积交织器和分组交织器；

所述卷积交织器用于：对n条通道数据流分别进行卷积交织得到n条第一数据流，每条所述第一数据流中连续的z个符号来自至少e个不同的码字，所述z为大于1的整数，a≤e≤F，e*k2≥F，每条所述第一数据流中连续的z个符号中最多k1/k2个符号来自同一个所述

码字；

所述分组交织器用于：对所述 n 条第一数据流中的每 K1 条第一数据流进行分组交织得到 S 条所述目标数据流，以得到共 m 条所述目标数据流，其中，所述 S 为大于或等于 1 的整数，m=S*n/K1，S≥k1/k2,所述 n 条第一数据流包括 K1 个第一数据流分组，同一第一数据流分组中每两条第一数据流的符号来自相同的码字，所述 K1 条第一数据流分别来自 K1 个第一数据流分组。

57.根据权利要求 56 所述的数据处理装置，其特征在于，所述卷积交织器具体用于：

根据 p 条延迟线对 1 条通道数据流进行延迟得到 1 条第一数据流，所述 p 为大于 1 的整数,p*a≥F/k2，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 d 个符号，所述 z=p*d，每条通道数据流中的符号按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 d 个符号且单次输出 d 个符号，1 条所述第一数据流中连续的 p*d 个符号包括每条延迟线输出的 d 个符号，所述 Q 为大于或等于 1 的整数，所述 d 为大于或等于 1 的整数，d≤a。

58.根据权利要求 57 所述的数据处理装置，其特征在于，所述 p 条延迟线中序号最大的延迟线包括 0 个存储单元，其中，$d(p*Q+1)\geq K4$。

59.根据权利要求 57 所述的数据处理装置，其特征在于，所述 p 条延迟线中序号最小的延迟线包括 0 个存储单元，其中，$d(p*Q-1)\geq K4$。

60.根据权利要求 57 至 59 中任一项所述的数据处理装置，其特征在于，参与分组交织的 K 条第一数据流包括第一符号矩阵，所述第一符号矩阵包括 K 行 B 列个符号，B=R*p*d，所述 R 为大于或等于 1 的整数，经过分组交织后的 S 条目标数据流包括第二符号矩阵，所述第二符号矩阵包括 S 行 F 列个符号，K*B=S*F；

所述第一符号矩阵中的符号至少来自 F 个不同码字，且所述第一符号矩阵中最多有 R*K1/K2 个符号来自相同的码字。

61.根据权利要求 60 所述的数据处理装置，其特征在于，所述第二符号矩阵中每行的 F 个符号来自所述第一符号矩阵中至少$\lfloor F/K2\rfloor$列，且所述$\lfloor F/K2\rfloor$列中每列最多取K2个符号，所述$\lfloor F/K2\rfloor$表示对 F/K2的商进行向上取整的整数；

所述第二符号矩阵中每行的 F 个符号包括所述第一符号矩阵中每行的至少$\lfloor F/K1\rfloor$个符号，所述$\lfloor F/K1\rfloor$表示对 F/K1 的商进行向下取整的整数，且所述第二符号矩阵中每行的 F 个符号包括所述第一符号矩阵中每行的最多$\lfloor F/K1\rfloor$个符号，所述$\lfloor F/K1\rfloor$表示对 F/K 的商进行向上取整的整数。

62.根据权利要求 60 或 61 所述的数据处理装置，其特征在于，所述第二符号矩阵的每一行中来自所述第一符号矩阵奇数列的符号分别位于所述第一符号矩阵的不同行，所述第二符号矩阵的每一行中来自所述第一符号矩阵偶数列的符号位于所述第一符号矩阵的不同行。

63.根据权利要求 60 至 62 中任一项所述的数据处理装置，其特征在于，所述第二符号矩阵的每一行中由延迟取值相同的延迟线输出的符号来自所述第一符号矩阵的不同行。

64.根据权利要求60至63中任一项所述的数据处理装置，其特征在于，所述第二符号矩阵的每一行的最多K3个符号来自所述第一符号矩阵的同一行，所述K3个符号中任意2个符号分别由延迟差值大于或等于2*Q*d的2条延迟线输出。

65.根据权利要求54或55所述的数据处理装置，其特征在于，所述卷积交织模块包括第一分组交织器、卷积交织器和第二分组交织器；

所述第一分组交织器用于：对n条通道数据流进行第一分组交织得到T条第一数据流，每条所述第一数据流中连续的C个符号至少来自E个不同的码字，T=n/K1，所述C为所述a的倍数，E≥K2*a；

所述卷积交织器用于：对所述T条第一数据流进行卷积交织得到T条第二数据流，每条所述第二数据流中连续的H个符号来自至少F个不同码字，F≥E，每条所述第二数据流中连续的H个符号最多K1/K2个符号来自相同码字；

所述第二分组交织器用于：对所述T条第二数据流中每条第二数据流进行第二分组交织得到S条目标数据流，以得到共m条目标数据流，m=T*S，S≥k1/K2。

66.根据权利要求65所述的数据处理装置，其特征在于，参与第一分组交织的n条通道数据流包括第三符号矩阵，所述第三符号矩阵包括n行A列个符号，所述A为所述a的倍数，经过第一分组交织后的T条第一数据流包括第四符号矩阵，所述第四符号矩阵包括T行C列个符号，所述T为所述n的约数，n*A=T*C；

所述第三符号矩阵的一列中每连续T个符号为符号子矩阵，所述第四符号矩阵中每一列的T个符号分别与所述第三符号矩阵中每个符号子矩阵一一对应。

67.根据权利要求66所述的数据处理装置，其特征在于，所述第三符号矩阵中的符号子矩阵按第一顺序排列，所述第三符号矩阵中每列的第1行到第n行包括按所述第一顺序排列的第1个符号子矩阵到第n/T个符号子矩阵，所述第三符号矩阵的相邻两列中前一列的第n/T个符号子矩阵到后一列的第1个符号子矩阵为按所述第一顺序排列的连续2个符号子矩阵，所述第四符号矩阵中第1列的T个符号来自所述第三符号矩阵中按所述第一顺序排列的第一个符号子矩阵，依次类推，直到所述第四符号矩阵中第C列的T个符号来自所述第三符号矩阵中按所述第一顺序排列的最后一个符号子矩阵；

或者，

所述第三符号矩阵中的符号子矩阵按第二顺序排列，所述第三符号矩阵中每T行的第1列到第A行包括按所述第二顺序排列的第1个符号子矩阵到第A个符号子矩阵，所述第三符号矩阵的相邻两个连续T行中前T行的第A个符号子矩阵到后T行的第1个符号子矩阵为按所述第二顺序排列的连续2个符号子矩阵，所述第四符号矩阵中第1列的T个符号来自所述第三符号矩阵中按所述第二顺序排列的第一个符号子矩阵，依次类推，直到所述第四符号矩阵中第C列的T个符号来自所述第三符号矩阵中按所述第二顺序排列的最后一个符号子矩阵。

68.根据权利要求65至67中任一项所述的数据处理装置，其特征在于，所述卷积交织器具体用于：

根据p条延迟线对1条第一数据流进行延迟得到1条第二数据流，所述p为大于1的整

数，p*E≥F,每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 C 个符号，p*C=H，每条第一数据流中的符号按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 C 个符号且单次输出 C 个符号，1 条所述第二数据流中连续的 p*C 个符号包括每条延迟线输出的 C 个符号，所述 Q 为大于或等于 1 的整数。

69. 根据权利要求 68 所述的数据处理装置，其特征在于，所述 p 条延迟线中序号最大的延迟线包括 0 个存储单元，其中，$C(p*Q+1)\geq K1*K4$。

70. 根据权利要求 68 所述的数据处理装置，其特征在于，所述 p 条延迟线中序号最小的延迟线包括 0 个存储单元，其中，$C(p*Q-1)\geq K1*K4$。

71. 根据权利要求 68 至 70 中任一项所述的数据处理装置，其特征在于，每条所述第二数据流包括 R 个符号集合，每个所述符号集合包括 p 个符号子集，每个所述符号子集包括 C 个符号，p 个符号子集分别由所述 p 条延迟线输出，每个符号集合中的符号来自至少 F 个不同码字，每条所述目标数据流包括 F 个符号，R*p*C=S*F，所述 R 为大于或等于的整数；

所述目标数据流中 F 个符号来自至少$\lceil F/(K2*a)\rceil$个不同的符号子集，且所述$\lceil F/(K2*a)\rceil$个不同的符号子集中每个符号子集最多取$K2*a$个符号，所述$\lceil F/(K2*a)\rceil$表示对$F/(K2*a)$的商进行向上取整的整数。

72. 根据权利要求 71 所述的数据处理装置，其特征在于，所述目标数据流中 F 个符号包括来自第一符号子集中的第一组符号和来自第二符号子集的第二组符号，所述第一符号子集和所述第二符号子集属于同一个符号集合，所述第一符号子集和所述第二符号子集分别由两条相邻的延迟线输出，所述第一符号子集中的符号和所述第二符号子集中的符号分别按次序排列，所述第一组符号在所述第一符号子集中的次序与所述第二组符号在所述第二符号子集中的次序不同；

所述目标数据流中 F 个符号包括来自第三符号子集中的第三组符号和来自第四符号子集的第四组符号，所述第三符号子集和所述第四符号子集属于不同的符号集合，所述第三符号子集和所述第四符号子集由同一条延迟线输出，所述第三符号子集中的符号和所述第四符号子集中的符号分别按次序排列，所述第三组符号在所述第三符号子集中的次序与所述第四组符号在所述第四符号子集中的次序不同。

73. 根据权利要求 71 或 72 所述的数据处理装置，其特征在于，每条所述目标数据流中 F 个符号最多有$\lceil F/R\rceil$个符号来自相同的符号集合，所述$\lceil F/R\rceil$表示对F/R的商进行向上取整的整数。

74. 根据权利要求 54 至 73 中任一项所述的数据处理装置，其特征在于，所述数据处理装置还包括编码器，得到共 m 条所述目标数据流之后，所述编码器用于：

对所述 m 条目标数据流分别进行第二 FEC 编码，所述第二 FEC 编码的信息位长度等于 F 个符号。

75. 一种数据处理方法，其特征在于，包括：

对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流，以得到共

m 条第一数据流，n=q*t，m=q*s，所述 n 为大于 1 的整数，所述 n 能被所述 q 整除，所述 q 为大于或等于 1 的整数，所述 t 为大于或等于 1 的整数，所述 s 为大于或等于 1 的整数，所述 n 条通道数据流都经过第一前向纠错 FEC 编码，所述第一 FEC 编码后的每 a 个码字分布在 b 条所述通道数据流中，所述 a≤所述 b≤所述 n，所述 n 能被所述 b 整除，所述 a 为大于或等于 1 的整数，每条通道数据流中每连续的 a 个符号来自 a 个不同的码字，且每条通道数据流中每连续的 L1 个符号来自至少 a 个不同的码字，L1=N*a/b,所述 N 表示所述码字的长度，所述 t 条通道数据流包括每条通道数据流中连续的 a 个符号共 t*a 个符号，所述 t*a 个符号包括每个符号中的 Δ 个比特共 D 个比特，D=Δ*t*a，所述 D 个比特在所述 s 条第一数据流中任一条第一数据流中连续，Δ=M/s,所述 M 表示一个符号包含的比特数；

对所述 m 条第一数据流分别进行卷积交织得到 m 条第二数据流。

76.根据权利要求 75 所述的数据处理方法，其特征在于，每条所述第一数据流中每连续的 d 个符号来自 v 个不同的码字,且每条所述第一数据流中每连续的 L2 个符号来自至少 v 个不同的码字，所述 v 能被所述 a 整除，L2=t/s*L1，d=D/M。

77.根据权利要求 75 或 76 所述的方法，其特征在于，所述 n=32，所述 n 条通道数据流中位于奇数通道的 16 条通道数据流来自相同的码字,所述 n 条通道数据流中位于偶数通道的 16 条通道数据流来自相同的码字，所述 n 条通道数据流中位于奇数通道的数据流和位于偶数通道的数据流来自不同的码字。

78.根据权利要求 77 所述的方法，其特征在于，所述 t=2，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 2*i 条通道数据流和第 2*i+1 条通道数据流进行分组交织得到 1 条第一数据流，0≤i＜16,所述第 2*i 条通道数据流中连续的 2 个符号和所述第 2*i+1 条通道数据流中连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

79.根据权利要求 78 所述的方法，其特征在于，所述分组交织得到的 1 条第一数据流中连续的 40 个比特中第j个比特来自第$2*i+\left\lfloor\frac{j}{\beta}\right\rfloor\%2$条通道数据流中连续 20 个比特中的第$\left\lfloor\left\lfloor\frac{j}{\beta}\right\rfloor/2\right\rfloor*\beta+j\%\beta$个比特，所述[x]表示向下取整，$0\le j<40$，所述β为 1,2,4,5,10 或 20。

80.根据权利要求 77 所述的方法，其特征在于，所述 t=2，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 2*i 条通道数据流和第 2*i+1 条通道数据流进行分组交织得到 1 条第一数据流，0≤i＜16，所述第 2*i 条通道数据流中第 j 组连续的β个比特与所述第 2*i+1 条通道数据流中第 j 组连续的β个比特在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述β为 1,2,4,5,10 或 20,所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

81.根据权利要求 77 所述的方法，其特征在于，所述 t=2，所述 s=2，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 2*i 条通道数据流和第 2*i+1 条通道数据流进行分组交织得到第 2*i 条第一数据流和第 2*i+1 条第一数据流，0≤i＜16，所述第 2*i 条通道数据流中连续的 2 个符号和所述第 2*i+1 条通道数据流中连续的 2 个符号共 4 个符号中每个符号的其中 5 个比特共 20 个比特在所述第 2*i 条第一数据流中连续，所述第 2*i 条第一数据流中每连续的 20 个比特来自 4 个不同的码字，所述第 2*i 条通道数据流中连续的 2 个符号和所述第 2*i+1 条通道数据流中连续的 2 个符号共 4 个符号中每个符号的另外 5 个比特共 20 个比特在所述第 2*i+1 条第一数据流中连续，所述第 2*i+1 条第一数据流中每连续的 20 个比特来自 4 个不同的码字。

82. 根据权利要求 81 所述的方法，其特征在于，第 2*i+g 条第一数据流中连续的 20 个比特中第f个比特来自第$2*i+\left\lfloor\frac{f}{10}\right\rfloor$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f\%10}{5}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

83. 根据权利要求 81 所述的方法，其特征在于，第 2*i+g 条第一数据流中连续的 20 个比特中第f个比特来自第$2*i+\left\lfloor\frac{f}{5}\right\rfloor\%2$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f}{10}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

84. 根据权利要求 77 所述的方法，其特征在于，所述 t=4，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流和第 4*i+3 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的连续 2 符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自至少 4 个不同的码字、且每连续的 272 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

85. 根据权利要求 77 所述的方法，其特征在于，所述 t=4，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流和第 4*i+3 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，所述 4 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自至少 4 个不同的码字、且每连续的 272 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

86. 根据权利要求 77 所述的方法，其特征在于，所述 t=4，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流和第 4*i+3 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的第 j 个符号共 4 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

87. 根据权利要求 77 所述的方法，其特征在于，所述 t=8，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 8*i 条通道数据流、第 8*i+1 条通道数据流、第 8*i+2 条通道数据流、第 8*i+3 条通道数据流、第 8*i+4 条通道数据流、第 8*i+5 条通道数据流、第 8*i+6 条通道数据流、第 8*i+7 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的连续 2 符号共 16 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号来自至少 4 个不同的码字、且每连续的 544 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 8 个符号、第 9 个符号、第 10 个符号和第 11 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 12 个符号、第 13 个符号、第 14 个符号和第 15 个符号分别来自不同的码字。

88. 根据权利要求 77 所述的方法，其特征在于，所述 t=8，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 8*i 条通道数据流、第 8*i+1 条通道数据流、第 8*i+2 条通道数据流、第 8*i+3 条通道数据流、第 8*i+4 条通道数据流、第 8*i+5 条通道数据流、第 8*i+6 条通道数据流、第 8*i+7 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号共 16 个符号在所述分组交织得到的 1 条第一数据流中连续，所述 8 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号来自至少 4 个不同的码字、且每连续的 544 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 8 个符号、第 9 个符号、第 10 个符号和第 11 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 12 个符号、第 13 个符号、第 14 个符号和第 15 个符号分别来自不同的码字。

89. 根据权利要求 77 所述的方法，其特征在于，所述 t=8，所述 s=1，对 n 条通道数据流

中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 8*i 条通道数据流、第 8*i+1 条通道数据流、第 8*i+2 条通道数据流、第 8*i+3 条通道数据流、第 8*i+4 条通道数据流、第 8*i+5 条通道数据流、第 8*i+6 条通道数据流、第 8*i+7 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的第 j 个符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

90. 根据权利要求 75 或 76 所述的方法，其特征在于，所述 n=32，所述 n 条通道数据流中排序在前的连续 16 条通道数据流来自相同的码字，所述 n 条通道数据流中排序在后的连续 16 条通道数据流来自相同的码字，所述 n 条通道数据流中排序在前的连续 16 条通道数据流和排序在后的连续 16 条通道数据流来自不同的码字。

91. 根据权利要求 90 所述的方法，其特征在于，所述 t=2，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 i 条通道数据流和第 i+16 条通道数据流进行分组交织得到 1 条第一数据流，0≤i＜16，所述第 i 条通道数据流中连续的 2 个符号和所述第 i+16 条通道数据流中连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

92. 根据权利要求 91 所述的方法，其特征在于，所述分组交织得到的 1 条第一数据流中连续的 40 个比特中第j个比特来自第$2*i+\left\lfloor \frac{j}{\beta} \right\rfloor \%2$条通道数据流中连续 20 个比特中的第$\left\lfloor \left\lfloor \frac{j}{\beta} \right\rfloor /2 \right\rfloor *\beta+j\%\beta$个比特，所述[x]表示向下取整，$0 \le j < 40$，所述β为 1, 2, 4, 5, 10 或 20。

93. 根据权利要求 90 所述的方法，其特征在于，所述 t=2，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 i 条通道数据流和第 i+16 条通道数据流进行分组交织得到 1 条第一数据流，0≤i＜16，所述第 i 条通道数据流中第 j 组连续的β个比特与所述第 i+16 条通道数据流中第 j 组连续的β个比特在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述β为 1, 2, 4, 5, 10 或 20，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

94. 根据权利要求 90 所述的方法，其特征在于，所述 t=2，所述 s=2，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 i 条通道数据流和第 i+16 条通道数据流进行分组交织得到第 2*i 条第一数据流和第 2*i+1 条第一数据流，0≤i＜16，所述第 i 条通道数据流中连续的 2 个符号和所述第 i+16 条通道数据流中连续的 2 个符号共 4 个符号中每个符号的其中 5 个比特共 20 个比特在所述第 2*i 条第一数据流中连续，所述第 2*i 条第一数据流中每连续的 20 个比特来自 4 个不同的码

字，所述第 i 条通道数据流中连续的 2 个符号和所述第 i+16 条通道数据流中连续的 2 个符号共 4 个符号中每个符号的另外 5 个比特共 20 个比特在所述第 2*i+1 条第一数据流中连续，所述第 2*i+1 条第一数据流中每连续的 20 个比特来自 4 个不同的码字。

95. 根据权利要求 94 所述的方法，其特征在于，第 2*i+g 条第一数据流中连续的 20 个比特中第f个比特来自第$i+16*\left\lfloor\frac{f}{10}\right\rfloor$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f\%10}{5}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

96. 根据权利要求 94 所述的方法，其特征在于，第 2*i+g 条第一数据流中连续的 20 个比特中第f个比特来自第$i+16*\left\lfloor\frac{f}{10}\right\rfloor$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f}{10}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

97. 根据权利要求 90 所述的方法，其特征在于，所述 t=4，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 2*i 条通道数据流、第 2*i+1 条通道数据流、第 2*i+16 条通道数据流和第 2*i+17 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的连续 2 符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自至少 4 个不同的码字、且每连续的 272 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

98. 根据权利要求 90 所述的方法，其特征在于，所述 t=4，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 2*i 条通道数据流、第 2*i+1 条通道数据流、第 2*i+16 条通道数据流和第 2*i+17 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，所述 4 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自至少 4 个不同的码字、且每连续的 272 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

99. 根据权利要求 90 所述的方法，其特征在于，所述 t=4，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 2*i 条通道数据流、第 2*i+1 条通道数据流、第 2*i+16 条通道数据流和第 2*i+17 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的第 j 个符号共 4 个符号在所述分组交织得到的 1 条第一数据

流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

100. 根据权利要求 90 所述的方法，其特征在于，所述 t=8，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流、第 4*i+16 条通道数据流、第 4*i+17 条通道数据流、第 4*i+18 条通道数据流和第 4*i+19 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的连续 2 符号共 16 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号来自至少 4 个不同的码字、且每连续的 544 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 8 个符号、第 9 个符号、第 10 个符号和第 11 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 12 个符号、第 13 个符号、第 14 个符号和第 15 个符号分别来自不同的码字。

101. 根据权利要求 90 所述的方法，其特征在于，所述 t=8，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流、第 4*i+16 条通道数据流、第 4*i+17 条通道数据流、第 4*i+18 条通道数据流和第 4*i+19 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号共 16 个符号在所述分组交织得到的 1 条第一数据流中连续，所述 8 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号来自至少 4 个不同的码字、且每连续的 544 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 8 个符号、第 9 个符号、第 10 个符号和第 11 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 12 个符号、第 13 个符号、第 14 个符号和第 15 个符号分别来自不同的码字。

102. 根据权利要求 90 所述的方法，其特征在于，所述 t=8，所述 s=1，对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流包括：

对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流、第 4*i+16 条通道数据流、第 4*i+17 条通道数据流、第 4*i+18 条通道数据流和

第 4*i+19 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第 i 数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的第 j 个符号共 8 个符号在所述分组交织得到的 1 条第 i 数据流中连续，j≥0，所述分组交织得到的 1 条第 i 数据流中每连续的 8 个符号至少来自 4 个不同的码字，所述分组交织得到的 1 条第 i 数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第 i 数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

103. 根据权利要求 75、76、77、78、79、80、81、82、83、90、91、92、93、94、95 或 96 所述的方法，其特征在于，对 1 条第一数据流进行卷积交织得到 1 条第二数据流包括：

根据 p 条延迟线对 1 条第一数据流进行延迟得到 1 条第二数据流，所述 p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 d 个符号，每条通道数据流中的符号按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 d 个符号且单次输出 d 个符号，1 条所述第二数据流中连续的 p*d 个符号包括每条延迟线输出的 d 个符号，所述 Q 为大于或等于 1 的整数。

104. 根据权利要求 103 所述的方法，其特征在于，所述 p 条延迟线中序号最大的延迟线包括 0 个存储单元，$d(p*Q+1) \geq L2$，L2=t/s*L1；

或者，

所述 p 条延迟线中序号最小的延迟线包括 0 个存储单元，$d(p*Q-1) \geq L2$，L2=t/s*L1。

105. 根据权利要求 103 或 104 所述的方法，其特征在于，对所述 m 条第一数据流分别进行卷积交织得到 m 条第二数据流之后，所述方法还包括：

对所述 m 条第二数据流分别进行第二 FEC 编码得到 m 条编码数据流，每条所述编码数据流中长度为 K 个符号的信息数据最多来自 K 个不同的码字，K≥p*d。

106. 根据权利要求 75、76、77、84、85、86、87、88、89、97、98、99、100、101 或 102 所述的方法，其特征在于，对 1 条第一数据流进行卷积交织得到 1 条第二数据流包括：

根据 p 条延迟线对 1 条第一数据流进行延迟得到 1 条第二数据流，所述 p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 4 个符号，每条通道数据流中的符号按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 4 个符号且单次输出 4 个符号，1 条所述第二数据流中连续的 p*4 个符号包括每条延迟线输出的 4 个符号，所述 Q 满足$4(p*Q-1) \geq 272$ 或者 $4(p*Q+1) \geq 272$ 或者 $4(p*Q-1 \geqslant 544$ 或者 $4p*Q+1 \geqslant 544$。

107. 根据权利要求 106 所述的方法，其特征在于，对所述 m 条第一数据流分别进行卷积交织得到 m 条第二数据流之后，所述方法还包括：

对所述 m 条第二数据流分别进行第二 FEC 编码得到 m 条编码数据流，每条所述编码数据流中长度为 K 个符号的信息数据最多来自 K 个不同的码字，K≥p*4。

108.根据权利要求 75 至 102 中任一项所述的方法，其特征在于，对 1 条第一数据流进行卷积交织得到 1 条第二数据流包括：

根据 p 条延迟线对 1 条第一数据流进行延迟得到 1 条第二数据流，所述 p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q；

每个存储单元用于存储 34 个比特，每条通道数据流中的比特按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 34 个比特且单次输出 34 个比特，1 条所述第二数据流中连续的 p*34 个比特包括每条延迟线输出的 34 个比特；或者，每个存储单元用于存储 68 个比特，每条通道数据流中的比特按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 68 个比特且单次输出 68 个比特，1 条所述第二数据流中连续的 p*68 个比特包括每条延迟线输出的 68 个比特。

109.根据权利要求 108 所述的方法，其特征在于，p=2，每个存储单元用于存储 68 个比特，或者，p=4，每个存储单元用于存储 34 个比特。

110.一种数据处理装置，其特征在于，包括：分组交织器和卷积交织器；

所述分组交织器用于：对 n 条通道数据流中的每 t 条通道数据流进行分组交织得到 s 条第一数据流，以得到共 m 条第一数据流，n=q*t，m=q*s，所述 n 为大于 1 的整数，所述 n 能被所述 q 整除，所述 q 为大于或等于 1 的整数，所述 t 为大于或等于 1 的整数，所述 s 为大于或等于 1 的整数，所述 n 条通道数据流都经过第一前向纠错 FEC 编码，所述第一 FEC 编码后的每 a 个码字分布在 b 条所述通道数据流中，所述 a≤所述 b≤所述 n，所述 n 能被所述 b 整除，所述 a 为大于或等于 1 的整数，每条通道数据流中每连续的 a 个符号来自 a 个不同的码字，且每条通道数据流中每连续的 L1 个符号来自至少 a 个不同的码字，L1=N*a/b，所述 N 表示所述码字的长度，所述 t 条通道数据流包括每条通道数据流中连续的 a 个符号共 t*a 个符号，所述 t*a 个符号包括每个符号中的 Δ 个比特共 D 个比特，D=Δ*t*a，所述 D 个比特在所述 s 条第一数据流中任一条第一数据流中连续，Δ=M/s，所述 M 表示一个符号包含的比特数；

所述卷积交织器用于：对所述 m 条第一数据流分别进行卷积交织得到 m 条第二数据流。

111.根据权利要求 110 所述的数据处理装置，其特征在于，每条所述第一数据流中每连续的 d 个符号来自 v 个不同的码字，且每条所述第一数据流中每连续的 L2 个符号来自至少 v 个不同的码字，所述 v 能被所述 a 整除，L2=t/s*L1，d=D/M。

112.根据权利要求 110 或 111 所述的数据处理装置，其特征在于，所述 n=32，所述 n 条通道数据流中位于奇数通道的 16 条通道数据流来自相同的码字，所述 n 条通道数据流中位于偶数通道的 16 条通道数据流来自相同的码字，所述 n 条通道数据流中位于奇数通道的数据流和位于偶数通道的数据流来自不同的码字。

113.根据权利要求 112 所述的数据处理装置，其特征在于，所述 t=2，所述 s=1，所述分组交织器具体用于：

对第 2*i 条通道数据流和第 2*i+1 条通道数据流进行分组交织得到 1 条第一数据流，0≤i＜16，所述第 2*i 条通道数据流中连续的 2 个符号和所述第 2*i+1 条通道数据流中连续的

2 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

114. 根据权利要求 113 所述的数据处理装置，其特征在于，所述分组交织得到的 1 条第一数据流中连续的 40 个比特中第j个比特来自第$2*i+\left\lfloor\frac{j}{\beta}\right\rfloor\%2$条通道数据流中连续 20 个比特中的第$\left\lfloor\left\lfloor\frac{j}{\beta}\right\rfloor/2\right\rfloor*\beta+j\%\beta$个比特，所述[x]表示向下取整，$0\le j<40$，所述β为 1, 2, 4, 5, 10 或 20。

115. 根据权利要求 112 所述的数据处理装置，其特征在于，所述 t=2，所述 s=1，所述分组交织器具体用于：

对第 2*i 条通道数据流和第 2*i+1 条通道数据流进行分组交织得到 1 条第一数据流，0≤i＜16，所述第 2*i 条通道数据流中第 j 组连续的β个比特与所述第 2*i+1 条通道数据流中第 j 组连续的β个比特在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述β为 1, 2, 4, 5, 10 或 20，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

116. 根据权利要求 112 所述的数据处理装置，其特征在于，所述 t=2，所述 s=2，所述分组交织器具体用于：

对第 2*i 条通道数据流和第 2*i+1 条通道数据流进行分组交织得到第 2*i 条第一数据流和第 2*i+1 条第一数据流，0≤i＜16，所述第 2*i 条通道数据流中连续的 2 个符号和所述第 2*i+1 条通道数据流中连续的 2 个符号共 4 个符号中每个符号的其中 5 个比特共 20 个比特在所述第 2*i 条第一数据流中连续，所述第 2*i 条第一数据流中每连续的 20 个比特来自 4 个不同的码字，所述第 2*i 条通道数据流中连续的 2 个符号和所述第 2*i+1 条通道数据流中连续的 2 个符号共 4 个符号中每个符号的另外 5 个比特共 20 个比特在所述第 2*i+1 条第一数据流中连续，所述第 2*i+1 条第一数据流中每连续的 20 个比特来自 4 个不同的码字。

117. 根据权利要求 116 所述的数据处理装置，其特征在于，第 2*i+g 条第一数据流中连续的 20 个比特中第f个比特来自第$2*i+\left\lfloor\frac{f}{10}\right\rfloor$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f\%10}{5}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

118. 根据权利要求 116 所述的数据处理装置，其特征在于，第 2*i+g 条第一数据流中连续的 20 个比特中第f个比特来自第$2*i+\left\lfloor\frac{f}{5}\right\rfloor\%2$ 条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f}{10}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

119. 根据权利要求 112 所述的数据处理装置，其特征在于，所述 t=4，所述 s=1，所述分组交织器具体用于：

对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流和第 4*i+3 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的连续 2 符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自至少 4 个不同的码字、且每连续的 272 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每

连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

120. 根据权利要求 112 所述的数据处理装置，其特征在于，所述 t=4，所述 s=1，所述分组交织器具体用于：

对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流和第 4*i+3 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，所述 4 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自至少 4 个不同的码字、且每连续的 272 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

121. 根据权利要求 112 所述的数据处理装置，其特征在于，所述 t=4，所述 s=1，所述分组交织器具体用于：

对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流和第 4*i+3 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的第 j 个符号共 4 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

122. 根据权利要求 112 所述的数据处理装置，其特征在于，所述 t=8，所述 s=1，所述分组交织器具体用于：

对第 8*i 条通道数据流、第 8*i+1 条通道数据流、第 8*i+2 条通道数据流、第 8*i+3 条通道数据流、第 8*i+4 条通道数据流、第 8*i+5 条通道数据流、第 8*i+6 条通道数据流、第 8*i+7 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的连续 2 符号共 16 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号来自至少 4 个不同的码字、且每连续的 544 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 8 个符号、第 9 个符号、第 10 个符号和第 11 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 12 个符号、第 13 个符号、第 14 个符号和第 15 个符号分别来自不同的码字。

123. 根据权利要求 112 所述的数据处理装置，其特征在于，所述 t=8，所述 s=1，所述分

组交织器具体用于：

对第 8*i 条通道数据流、第 8*i+1 条通道数据流、第 8*i+2 条通道数据流、第 8*i+3 条通道数据流、第 8*i+4 条通道数据流、第 8*i+5 条通道数据流、第 8*i+6 条通道数据流、第 8*i+7 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号共 16 个符号在所述分组交织得到的 1 条第一数据流中连续，所述 8 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号来自至少 4 个不同的码字、且每连续的 544 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 8 个符号、第 9 个符号、第 10 个符号和第 11 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 12 个符号、第 13 个符号、第 14 个符号和第 15 个符号分别来自不同的码字。

124.根据权利要求 112 所述的数据处理装置，其特征在于，所述 t=8，所述 s=1，所述分组交织器具体用于：

对第 8*i 条通道数据流、第 8*i+1 条通道数据流、第 8*i+2 条通道数据流、第 8*i+3 条通道数据流、第 8*i+4 条通道数据流、第 8*i+5 条通道数据流、第 8*i+6 条通道数据流、第 8*i+7 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的第 j 个符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

125.根据权利要求 110 或 111 所述的数据处理装置，其特征在于，所述 n=32，所述 n 条通道数据流中排序在前的连续 16 条通道数据流来自相同的码字，所述 n 条通道数据流中排序在后的连续 16 条通道数据流来自相同的码字，所述 n 条通道数据流中排序在前的连续 16 条通道数据流和排序在后的连续 16 条通道数据流来自不同的码字。

126.根据权利要求 125 所述的数据处理装置，其特征在于，所述 t=2，所述 s=1，所述分组交织器具体用于：

对第 i 条通道数据流和第 i+16 条通道数据流进行分组交织得到 1 条第一数据流，0≤i＜16，所述第 i 条通道数据流中连续的 2 个符号和所述第 i+16 条通道数据流中连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

127. 根据权利要求 126 所述的数据处理装置，其特征在于，所述分组交织得到的 1 条第一数据流中连续的 40 个比特中第j个比特来自第$2*i+\left\lfloor\frac{j}{\beta}\right\rfloor\%2$条通道数据流中连续 20 个比特中的第$\left\lfloor\left\lfloor\frac{j}{\beta}\right\rfloor/2\right\rfloor*\beta+j\%\beta$个比特，所述[x]表示向下取整，$0\le j<40$，所述β为 1, 2, 4, 5, 10 或 20。

128. 根据权利要求 125 所述的数据处理装置，其特征在于，所述 t=2，所述 s=1，所述分组交织器具体用于：

对第 i 条通道数据流和第 i+16 条通道数据流进行分组交织得到 1 条第一数据流，0≤i＜16，所述第 i 条通道数据流中第 j 组连续的β个比特与所述第 i+16 条通道数据流中第 j 组连续的β个比特在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述β为 1, 2, 4, 5, 10 或 20，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

129. 根据权利要求 125 所述的数据处理装置，其特征在于，所述 t=2，所述 s=2，所述分组交织器具体用于：

对第 i 条通道数据流和第 i+16 条通道数据流进行分组交织得到第 2*i 条第一数据流和第 2*i+1 条第一数据流，0≤i＜16，所述第 i 条通道数据流中连续的 2 个符号和所述第 i+16 条通道数据流中连续的 2 个符号共 4 个符号中每个符号的其中 5 个比特共 20 个比特在所述第 2*i 条第一数据流中连续，所述第 2*i 条第一数据流中每连续的 20 个比特来自 4 个不同的码字，所述第 i 条通道数据流中连续的 2 个符号和所述第 i+16 条通道数据流中连续的 2 个符号共 4 个符号中每个符号的另外 5 个比特共 20 个比特在所述第 2*i+1 条第一数据流中连续，所述第 2*i+1 条第一数据流中每连续的 20 个比特来自 4 个不同的码字。

130. 根据权利要求 129 所述的数据处理装置，其特征在于，第 2*i+g 条第一数据流中连续的 20 个比特中第f个比特来自第$i+16*\left\lfloor\frac{f}{10}\right\rfloor$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f\%10}{5}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

131. 根据权利要求 129 所述的数据处理装置，其特征在于，第 2*i+g 条第一数据流中连续的 20 个比特中第f个比特来自第$i+16*\left\lfloor\frac{f}{10}\right\rfloor$条通道数据流中连续 20 个比特中的第$5*g+\left\lfloor\frac{f}{10}\right\rfloor*10+f\%5$个比特，$0\le f<20$，$0\le g<2$。

132. 根据权利要求 125 所述的数据处理装置，其特征在于，所述 t=4，所述 s=1，所述分组交织器具体用于：

对第 2*i 条通道数据流、第 2*i+1 条通道数据流、第 2*i+16 条通道数据流和第 2*i+17 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的连续 2 符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自至少 4 个不同的码字、且每连续的 272 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

133.根据权利要求 125 所述的数据处理装置，其特征在于，所述 t=4，所述 s=1，所述分组交织器具体用于：

对第 2*i 条通道数据流、第 2*i+1 条通道数据流、第 2*i+16 条通道数据流和第 2*i+17 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，所述 4 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号来自至少 4 个不同的码字、且每连续的 272 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

134.根据权利要求 125 所述的数据处理装置，其特征在于，所述 t=4，所述 s=1，所述分组交织器具体用于：

对第 2*i 条通道数据流、第 2*i+1 条通道数据流、第 2*i+16 条通道数据流和第 2*i+17 条通道数据流共 4 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤7，所述 4 条通道数据流中每条通道数据流包括的第 j 个符号共 4 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 4 个符号来自 4 个不同的码字。

135.根据权利要求 125 所述的数据处理装置，其特征在于，所述 t=8，所述 s=1，所述分组交织器具体用于：

对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流、第 4*i+16 条通道数据流、第 4*i+17 条通道数据流、第 4*i+18 条通道数据流和第 4*i+19 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的连续 2 符号共 16 个符号在所述分组交织得到的 1 条第一数据流中连续，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号来自至少 4 个不同的码字、且每连续的 544 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 8 个符号、第 9 个符号、第 10 个符号和第 11 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 12 个符号、第 13 个符号、第 14 个符号和第 15 个符号分别来自不同的码字。

136.根据权利要求 125 所述的数据处理装置，其特征在于，所述 t=8，所述 s=1，所述分组交织器具体用于：

对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条

通道数据流、第 4*i+16 条通道数据流、第 4*i+17 条通道数据流、第 4*i+18 条通道数据流和第 4*i+19 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号共 16 个符号在所述分组交织得到的 1 条第一数据流中连续，所述 8 条通道数据流中每条通道数据流包括的第 j 组连续的 2 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号来自至少 4 个不同的码字、且每连续的 544 个符号来自至少 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 8 个符号、第 9 个符号、第 10 个符号和第 11 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 16 个符号的第 12 个符号、第 13 个符号、第 14 个符号和第 15 个符号分别来自不同的码字。

137. 根据权利要求 125 所述的数据处理装置，其特征在于，所述 t=8，所述 s=1，所述分组交织器具体用于：

对第 4*i 条通道数据流、第 4*i+1 条通道数据流、第 4*i+2 条通道数据流、第 4*i+3 条通道数据流、第 4*i+16 条通道数据流、第 4*i+17 条通道数据流、第 4*i+18 条通道数据流和第 4*i+19 条通道数据流共 8 条通道数据流进行分组交织得到 1 条第一数据流，0≤i≤3，所述 8 条通道数据流中每条通道数据流包括的第 j 个符号共 8 个符号在所述分组交织得到的 1 条第一数据流中连续，j≥0，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号至少来自 4 个不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 0 个符号、第 1 个符号、第 2 个符号和第 3 个符号分别来自不同的码字，所述分组交织得到的 1 条第一数据流中每连续的 8 个符号中的第 4 个符号、第 5 个符号、第 6 个符号和第 7 个符号分别来自不同的码字。

138. 根据权利要求 110、111、112、113、114、115、116、117、118、125、126、127、128、129、130 或 131 所述的数据处理装置，其特征在于，所述卷积交织器具体用于：

根据 p 条延迟线对 1 条第一数据流进行延迟得到 1 条第二数据流，所述 p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 d 个符号，每条通道数据流中的符号按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 d 个符号且单次输出 d 个符号，1 条所述第二数据流中连续的 p*d 个符号包括每条延迟线输出的 d 个符号，所述 Q 为大于或等于 1 的整数。

139. 根据权利要求 138 所述的数据处理装置，其特征在于，所述 p 条延迟线中序号最大的延迟线包括 0 个存储单元，$d(p*Q+1) \geq L2$，L2=t/s*L1；

或者，

所述 p 条延迟线中序号最小的延迟线包括 0 个存储单元，$d(p*Q-1) \geq L2$，L2=t/s*L1。

140.根据权利要求 138 或 139 所述的数据处理装置，其特征在于，所述数据处理装置还包括编码器，得到 m 条第二数据流之后，所述编码器用于：

对所述 m 条第二数据流分别进行第二 FEC 编码得到 m 条编码数据流，每条所述编码数据流中长度为 K 个符号的信息数据最多来自 K 个不同的码字，K≥p*d。

141.根据权利要求 110、111、112、119、120、121、122、123、124、132、133、134、135、136 或 137 所述的数据处理装置，其特征在于，所述卷积交织器具体用于：

根据 p 条延迟线对 1 条第一数据流进行延迟得到 1 条第二数据流，所述 p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 4 个符号，每条通道数据流中的符号按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 4 个符号且单次输出 4 个符号，1 条所述第二数据流中连续的 p*4 个符号包括每条延迟线输出的 4 个符号，所述 Q 满足$4(p*Q-1)\geq 272$ 或者 $4(p*Q+1)\geq 272$ 或者 $4(p*Q-1\geqslant 544$ 或者 $4p*Q+1\geqslant 544$。

142.根据权利要求 141 所述的数据处理装置，其特征在于，所述数据处理装置还包括编码器，得到 m 条第二数据流之后，所述编码器用于：

对所述 m 条第二数据流分别进行第二 FEC 编码得到 m 条编码数据流，每条所述编码数据流中长度为 K 个符号的信息数据最多来自 K 个不同的码字，K≥p*4。

143.根据权利要求 110 至 137 中任一项所述的数据处理装置，其特征在于，所述卷积交织器具体用于：

根据 p 条延迟线对 1 条第一数据流进行延迟得到 1 条第二数据流，所述 p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q；

每个存储单元用于存储 34 个比特，每条通道数据流中的比特按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 34 个比特且单次输出 34 个比特，1 条所述第二数据流中连续的 p*34 个比特包括每条延迟线输出的 34 个比特；或者，每个存储单元用于存储 68 个比特，每条通道数据流中的比特按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 68 个比特且单次输出 68 个比特，1 条所述第二数据流中连续的 p*68 个比特包括每条延迟线输出的 68 个比特。

144.根据权利要求 143 所述的数据处理装置，其特征在于，p=2，每个存储单元用于存储 68 个比特，或者，p=4，每个存储单元用于存储 34 个比特。

145.一种数据处理方法，其特征在于，包括：

根据 p 条延迟线对 n 条通道数据流分别进行延迟得到 n 条第一数据流，其中，所述 n 条通道数据流都经过第一前向纠错 FEC 编码，所述 p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 U 个比特，每条通道数据流中的比特按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 U 个比特且单次输出 U

个比特，1 条所述第二数据流中连续的 p*U 个比特包括每条延迟线输出的 U 个比特，所述 Q 为大于或等于 1 的整数，所述 U 为大于或等于 1 的整数；

对所述 n 条第一数据流分别进行第二 FEC 编码得到 n 条第二数据流，经过所述第二 FEC 编码后的所述第二数据流中每个码字的信息数据为所述 p 条延迟线单次输出的 p*U 个比特。

146. 根据权利要求 145 所述的方法，其特征在于，p*U=120、136 或 160。

147. 一种数据处理装置，其特征在于，包括：卷积交织器和编码器；

所述卷积交织器用于：根据 p 条延迟线对 n 条通道数据流分别进行延迟得到 n 条第一数据流，其中，所述 n 条通道数据流都经过第一前向纠错 FEC 编码，所述 p 为大于 1 的整数，每条延迟线包括的存储单元数量各不相同，存储单元数量最小的延迟线包括 0 个存储单元，每相邻两条延迟线的存储单元数量的差值为 Q，每个存储单元用于存储 U 个比特，每条通道数据流中的比特按照所述 p 条延迟线的序号依次输入所述 p 条延迟线，每条延迟线单次输入 U 个比特且单次输出 U 个比特，1 条所述第二数据流中连续的 p*U 个比特包括每条延迟线输出的 U 个比特，所述 Q 为大于或等于 1 的整数，所述 U 为大于或等于 1 的整数；

所述编码器用于：对所述 n 条第一数据流分别进行第二 FEC 编码得到 n 条第二数据流，经过所述第二 FEC 编码后的所述第二数据流中每个码字的信息数据为所述 p 条延迟线单次输出的 p*U 个比特。

148. 根据权利要求 147 所述的数据处理装置，其特征在于，p*U=120、136 或 160。