(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 110402545 A
(43)申请公布日 2019.11.01

(21)申请号 201880016811.7

(22)申请日 2018.02.06

(30)优先权数据
1702341.7 2017.02.13 GB

(85)PCT国际申请进入国家阶段日
2019.09.09

(86)PCT国际申请的申请数据
PCT/GB2018/050332 2018.02.06

(87)PCT国际申请的公布数据
WO2018/146462 EN 2018.08.16

(71)申请人 阿塞勒康姆有限公司
地址 英国南安普顿市

(72)发明人 R.蒙德 M.布雷扎 L.向

(74)专利代理机构 北京市柳沈律师事务所 11105
代理人 万里晴

(51)Int.Cl.
H03M 13/39(2006.01)
H03M 13/27(2006.01)
H03M 13/29(2006.01)

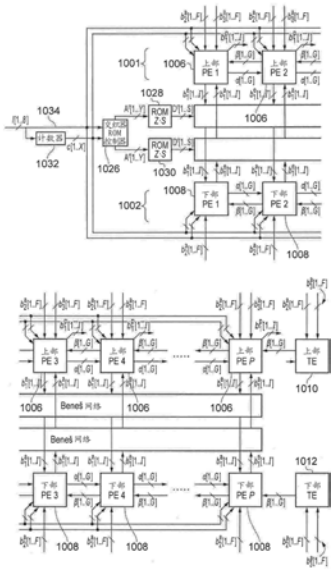
权利要求书5页 说明书39页 附图20页

(54)发明名称

利用非均匀窗口大小的并行TURBO解码

(57)摘要

turbo解码器电路执行turbo解码处理,以从接收到的信号中恢复数据符号的帧,该数据符号的帧包括该帧的每个数据符号的软判决值。该帧的数据符号已经用turbo编码器编码,该turbo编码器包括每个能够由网格表示的上部卷积编码器和下部卷积编码器、以及在上部卷积编码器和下部卷积编码器之间交织编码数据的交织器。turbo解码器电路包括时钟、用于交织软判决值的可配置网络电路、上部解码器和下部解码器。上部和下部解码器中的每一个包括处理元件,该处理元件被配置为在一系列连续时钟循环期间,从可配置网络电路迭代地接收与数据符号有关的先验软判决值,该先验软判决值与表示上部或下部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联。处理元件使用先验软判决值执行与该窗口相关联的并行计算,以生成与数据符号有关的对应非本征软判决值。可配置网络电路包括网络控制器电路,该网络控制器电路在连续时钟循环期间迭代地控制可配置网络电路的配置,以通过交织由下部解码器提供的非本征软判决值来为上部解码器提供先验软判决值,并且通过交织由上部解码器提供的非本征软判决值来为下部解码器提供先验软判决值。由网络控制器控制的可配置网络电路执行的交织是根据预定调度的,这在一个或多个连续时钟循环的不同循环提供先验软判决值,以避免在相同时钟循环期间向上部或下部解码器的相同处理元件提供不同先验软判决值之间的竞争。因此,处理元件可以具有包括网格的多个级的窗口大小,使得解码器可以被配置有任意数量的处理元件,使得解码器电路成为任意并行turbo解码器。



CN 110402545 A

1.一种turbo解码器电路,用于执行turbo解码处理以从接收到的信号中恢复数据符号的帧,所述数据符号的帧包括所述帧的每个数据符号的一个或多个奇偶校验和/或系统软判决值,所述帧的数据符号已经用turbo编码器编码,所述turbo编码器包括每个能够由网格表示的上部卷积编码器和下部卷积编码器、以及在所述上部卷积编码器和下部卷积编码器之间交织所述数据符号的交织器,所述turbo解码器电路包括:

时钟,

可配置网络电路,被配置为交织软判决值,

上部解码器,包括与所述上部卷积编码器相关联的多个上部处理元件,所述上部解码器的处理元件中的每一个处理元件被配置为,在一系列连续时钟循环期间,从所述可配置网络电路迭代地接收与数据符号有关的先验软判决值,其中所述先验软判决值与表示上部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联,以使用所述先验软判决值执行与该窗口相关联的并行计算以便生成与所述数据符号有关的对应非本征软判决值,以及被配置为向所述可配置网络电路提供非本征软判决值,所述上部解码器的至少一个处理元件被配置为相对所述上部解码器的至少一个其他处理元件执行与不同数量的网格级相关联的窗口的计算,以及

下部解码器,包括与所述下部卷积编码器相关联的多个下部处理元件,所述下部解码器的处理元件中的每一个处理元件被配置为,在所述系列连续时钟循环期间,从所述可配置网络电路迭代地接收与数据符号有关的先验软判决值,其中所述先验软判决值与表示下部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联,以使用所述先验软判决值执行与该窗口相关联的并行计算以便生成与所述数据符号有关的对应非本征软判决值,以及被配置为向所述可配置网络电路提供非本征软判决值,所述下部解码器的至少一个处理元件被配置为相对所述下部解码器的至少一个其他处理元件执行与不同数量的网格级相关联的窗口的计算,

其中所述可配置网络电路包括网络控制器电路,所述网络控制器电路在所述连续时钟循环期间迭代地控制可配置网络电路的配置,以通过交织由所述下部解码器提供的非本征软判决值来为所述上部解码器提供先验软判决值,以及通过交织由所述上部解码器提供的非本征软判决值来为所述下部解码器提供先验软判决值,其中由所述网络控制器控制的所述可配置网络电路执行的交织是根据预定调度的,这在一个或多个连续时钟循环的不同循环提供先验软判决值,以避免在相同时钟循环期间向上部或下部解码器的相同处理元件提供不同先验软判决值之间的竞争。

2.如权利要求1所述的turbo解码器电路,其中,所述上部解码器和所述下部解码器每个的处理元件被配置为从存储器读取先验软判决值,以及在执行计算之后将非本征软判决值写入存储器,并且所述可配置网络电路被配置为从存储器读取所述非本征软判决值,以及将先验软判决值写入存储器,并且由所述可配置网络根据所述预定调度对所述非本征软判决值中的一个或多个非本征软判决值的读取相对于由所述处理元件对所述一个或多个非本征软判决值的写入延迟一个或多个时钟循环。

3.如权利要求1所述的turbo解码器电路,其中,所述上部解码器和所述下部解码器每个的处理元件被配置为从存储器读取先验软判决值,以及在执行计算之后将非本征软判决值写入存储器,并且所述可配置网络电路被配置为从存储器读取所述非本征软判决值,以

及将先验软判决值写入存储器，以及由所述可配置网络根据所述预定调度对所述非本征软判决值中的一个或多个非本征软判决值的读取或者由所述处理元件对所述一个或多个非本征软判决值的写入中的至少一个被跳过。

4.如权利要求1所述的turbo解码器电路，其中，所述上部解码器或下部解码器中的处理元件的数量不是网格级的数量的整数倍。

5.如权利要求1所述的turbo解码器，其中，由所述处理元件处理的每个窗口内网格级的最小和最大数量之间的差在上部和下部解码器中的任一个中是一。

6.如权利要求1所述的turbo解码器电路，其中，所述窗口中的每个窗口包括由彼此相邻的处理元件处理的相同数量的网格级。

7.如权利要求1所述的turbo解码器电路，其中，所述上部和下部解码器包括相同数量的处理元件，并且所述上部解码器的每个处理元件如所述下部解码器的对应处理元件一样对包括对应网格级的窗口执行计算。

8.如权利要求1所述的turbo解码器，其中，所述处理元件和所述交织的处理调度根据相同数量的时钟循环是周期性的，每次迭代表示相同调度的周期。

9.如权利要求8所述的turbo解码器，其中，所述周期由所述上部或下部解码器中的任一个窗口中的最大网格级加上减少用于根据所述预定调度跳过以避免竞争的需求而所需的非负整数来给出。

10.如权利要求1所述的turbo解码器电路，其中，所述处理元件中的每一个处理元件被配置为根据周期性调度执行并行计算，并且每个周期包括第一子周期和第二子周期，所述第一子周期包括周期中的一个或多个第一时钟循环，所述第二子周期包括周期中的剩余循环，其中，在第一子周期期间，每个窗口的处理包括在包括窗口中的前一个或多个网格级的第一子窗口、或者包括窗口中的后一个或多个网格级的第二子窗口内的前向递归和后向递归后向递归，且在第二子周期期间，所述处理元件中的每一个处理元件被配置为在包括窗口中的剩余网格级的第一和第二子窗口中的另一个内执行前向递归和后向递归。

11.如权利要求10所述的turbo解码器电路，其中，第一和第二子周期中的一个包括在周期中的时钟循环的半向下舍入，第一和第二子周期中的另一个包括在周期中的时钟循环的剩余半向上舍入，并且在包括时钟循环的半向下舍入的第一子周期和第二子周期中的一个子周期期间，每个处理元件对包括窗口中的网格级的半向下舍入的第一或第二子窗口执行并行计算，并且在包括时钟循环的半向上舍入的第一子周期和第二子周期的另一个子周期期间，所述处理元件对包括窗口中的网格级的半向上舍入的第一或第二子窗口执行计算。

12.如权利要求11所述的turbo解码器电路，其中，所述处理元件被配置为在与子窗口内的完全前向递归和子窗口内的完全后向递归相关联的子周期内对所述子窗口执行计算，并且在执行所述完全前向递归和所述完全后向递归之后，任何剩余的时钟循环被所述处理元件用来执行与后续前向递归和后续后向递归的至少一部分相关联的计算。

13.如权利要求12所述的turbo解码器电路，其中，在所述第一子周期期间，所述上部解码器的处理元件被配置为执行与第一或第二子窗口中的相同的一个相关联的计算，并且所述下部解码器的处理元件被配置为执行与第一或第二子窗口中的另一个相关联的计算，并且

在所述第二子周期期间，所述上部解码器的处理元件被配置为执行与在所述第一子周期期间未被所述处理元件处理的第一或第二子窗口相关联的计算，并且所述下部解码器的处理元件被配置为执行与在所述第一子周期期间未被所述处理元件处理的第一或第二子窗口中的另一个相关联的计算。

14.如权利要求10所述的turbo解码器电路，其中，所述前向递归根据执行与前向方向上的每个连续网格级相关联的计算的调度来生成与多个网格状态相对应的多个前向状态度量，并且所述后向递归根据执行与后向方向上的每个连续网格级相关联的计算的调度来生成与多个网格状态相对应的多个后向状态度量，并且所述前向递归根据用于所述前向递归的调度将所述前向状态度量存储在存储器中，或者所述后向递归根据用于所述后向递归的调度将所述后向状态度量存储在存储器中，并且所述前向递归或所述后向递归中的另一个从所述存储器加载存储的前向或后向状态度量，并组合前向和后向状态度量以根据用于所述前向或后向递归的调度来计算非本征软判决值。

15.如权利要求10所述的turbo解码器电路，其中，由所述处理元件根据前向递归和后向递归而执行的计算包括：接收与相邻网格级有关的前向或后向状态度量，将所述前向或后向状态度量与所述数据符号的先验、奇偶校验和系统软判决值进行组合，以及生成与另一相邻网格级有关的前向或后向状态度量，其中所接收的前向或后向状态度量在与所述先验、奇偶校验和系统软判决值组合之前被归一化。

16.如权利要求15所述的turbo解码器电路，其中，所述处理元件被配置为根据两步流水线来生成非本征软判决值，所述两步流水线包括第一步以及第二步，所述第一步将前向和后向状态度量彼此组合并与所述奇偶校验软判决值组合以形成中间变量，所述第二步将所述中间变量彼此组合、缩放所述中间变量的组合并将缩放的所述中间变量的组合与所述系统软判决值组合，并且流水线的所述两步在两个连续的时钟循环期间被执行，并且由流水线的步骤施加的延迟被包容在由所述可配置网络的预定调度施加的延迟中以避免竞争。

17.如权利要求1所述的turbo解码器电路，其中，所述帧中的数据符号的数量是可变的，并且由上部和下部解码器执行的计算的每个窗口的网格级的数量是相对于帧长度和所述上部和下部解码器的处理元件的数量来确定的。

18.一种从接收到的信号中恢复数据符号的帧的turbo解码方法，所述数据符号的帧包括所述帧的每个数据符号的一个或多个奇偶校验和/或系统软判决值，所述帧的数据符号已经用turbo编码器编码，所述turbo编码器包括每个能够由网格表示的上部卷积编码器和下部卷积编码器、以及交织已经在上部卷积编码器和下部卷积编码器之间交织的编码数据的交织器，所述方法包括：

使用包括与上部卷积编码器相关联的多个上部处理元件的上部解码器，通过如下步骤来执行前向和后向迭代递归处理：

在一系列连续时钟循环期间，在所述上部解码器的处理元件中的每一个处理元件处从可配置网络电路迭代地接收与数据符号有关的先验软判决值，其中所述先验软判决值与表示上部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联，

由所述处理单元中的每一个处理元件使用先验软判决值执行与该窗口相关联的并行计算，以便生成与所述数据符号有关的对应非本征软判决值，所述上部解码器的至少一个处理元件相对上部解码器的至少一个其他处理元件执行与不同数量的网格级相关联的窗

口的计算，

向可配置网络电路提供非本征软判决值，以及

使用包括与下部卷积编码器相关联的多个下部处理元件的下部解码器，通过如下步骤来执行前向和后向迭代递归处理：

在所述系列连续时钟循环期间，在所述下部解码器的处理元件中的每一个处理元件处从可配置网络电路迭代地接收与数据符号有关的先验软判决值，其中所述先验软判决值与表示下部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联，

由所述处理单元中的每一个处理元件使用先验软判决值执行与该窗口相关联的并行计算，以便生成与所述数据符号有关的对应非本征软判决值，所述下部解码器的至少一个处理元件相对所述下部解码器的至少一个其他处理元件执行与不同数量的网格级相关联的窗口的计算，

向可配置网络电路提供非本征软判决值，

在连续时钟循环期间，迭代地控制可配置网络电路的配置，以通过交织由下部解码器提供的非本征软判决值来为上部解码器提供先验软判决值，并且通过交织由上部解码器提供的非本征软判决值来为下部解码器提供先验软判决值，其中，由网络控制器控制的可配置网络电路执行的交织是根据预定调度的，这在一个或多个连续时钟循环的不同循环提供先验软判决值，以避免在相同时钟循环期间向上部或下部解码器的相同处理元件提供不同先验软判决值之间的竞争。

19.一种接收器，用于检测和恢复已经用turbo码编码的数据符号的帧，所述接收器包括：

检测电路，用于检测携带所述数据符号的帧的接收到的信号，每个数据符号的帧包括所述帧的每个数据符号的一个或多个奇偶校验和/或系统软判决值，每个帧的数据符号已经用turbo编码器编码，所述turbo编码器包括每个能够由网格表示的上部卷积编码器和下部卷积编码器、以及交织已经在上部卷积编码器和下部卷积编码器之间交织的编码数据的交织器，以及

turbo解码器电路，用于执行turbo解码处理以从所述接收到的信号中恢复所述数据符号的帧中的每一个，所述turbo解码器电路包括：

时钟，

可配置网络电路，被配置为交织软判决值，

上部解码器，包括与所述上部卷积编码器相关联的多个上部处理元件，所述上部解码器的处理元件中的每一个处理元件被配置为，在一系列连续时钟循环期间，从所述可配置网络电路迭代地接收与数据符号有关的先验软判决值，其中所述先验软判决值与表示上部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联，以使用所述先验软判决值执行与该窗口相关联的并行计算以便生成与所述数据符号有关的对应非本征软判决值，以及被配置为向所述可配置网络电路提供非本征软判决值，所述上部解码器的至少一个处理元件被配置为相对所述上部解码器的至少一个其他处理元件执行与不同数量的网格级相关联的窗口的计算，以及

下部解码器，包括与所述下部卷积编码器相关联的多个下部处理元件，所述下部解码器的处理元件中的每一个处理元件被配置为，在所述系列连续时钟循环期间，从所述可配

置网络电路迭代地接收与数据符号有关的先验软判决值，其中所述先验软判决值与表示下部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联，以使用所述先验软判决值执行与该窗口相关联的并行计算以便生成与所述数据符号有关的对应非本征软判决值，以及被配置为向所述可配置网络电路提供非本征软判决值，所述下部解码器的至少一个处理元件被配置为相对所述下部解码器的至少一个其他处理元件执行与不同数量的网格级相关联的窗口的计算，

其中所述可配置网络电路包括网络控制器电路，所述网络控制器电路在所述连续时钟循环期间迭代地控制所述可配置网络电路的配置，以通过交织由所述下部解码器提供的非本征软判决值来为所述上部解码器提供先验软判决值，以及通过交织由所述上部解码器提供的非本征软判决值来为所述下部解码器提供先验软判决值，其中由所述网络控制器控制的所述可配置网络电路执行的交织是根据预定调度的，这在一个或多个连续时钟循环的不同循环提供先验软判决值，以避免在相同时钟循环期间向上部或下部解码器的相同处理元件提供不同先验软判决值之间的竞争。

20.如权利要求19所述的接收器，其中，每个帧中的数据符号的数量从一个到另一个动态地变化。

21.一种形成无线通信网络的无线电接入网络的一部分的基础设施装备，所述基础设施装备包括如权利要求19所述的接收器。

22.一种用于利用无线通信网络发送或接收数据的通信设备，所述通信设备包括如权利要求19所述的接收器。

利用非均匀窗口大小的并行TURBO解码

技术领域

[0001] 本公开涉及用于执行turbo检测处理以从接收到的信号中恢复数据符号的帧的检测电路，该数据符号的帧包括该帧的每个数据符号的一个或多个奇偶校验和/或系统软判决值。该帧的数据符号已经用turbo编码器进行编码，该turbo编码器包括上部卷积编码器和下部卷积编码器，每个卷积编码器可以由具有多个网格状态的网格表示。

[0002] 因此，本公开的实施例可以提供被配置为使用turbo解码器来恢复数据符号的帧的接收器和用于解码被turbo编码的数据的方法。在一个示例中，数据符号是比特。

[0003] 本申请要求对英国专利申请1702341.7的巴黎公约优先权，其内容通过引用结合于此。

背景技术

[0004] 在过去的二十年里，无线通信已经被受益于迭代解码算法的信道码而彻底改变了。例如，长期演进(Long Term Evolution,LTE)[1]和WiMAX[2]蜂窝电话标准采用turbo码[3]，该turbo码包括两个卷积码的级联。传统上，对数巴赫-科克-耶利内克-拉维夫(Logarithmic Bahl-Cocke-Jelinek-Raviv,对数-BCJR)算法[4]被用于马尔可夫链的迭代解码，该马尔可夫链由这些卷积码施加在编码比特上。同时，无线局域网(Wireless Local Area Networks,WLAN)的Wi-Fi标准[5]已经采用低密度奇偶校验(Low Density Parity Check,LDPC)码[6]，其可以基于最小和算法[7]操作。由于它们强大的纠错能力，这些复杂的信道编码已经促进了以紧密接近无线信道的容量的传输吞吐量的可靠通信。然而，如果要求实时操作，可实现的传输吞吐量受到迭代解码算法的处理吞吐量的限制。此外，迭代解码算法的处理延迟对端到端延迟施加了限制。这尤其重要，因为多千兆比特传输吞吐量和超低端到端延迟可能预期是下一代无线通信标准[8]的目标。因此，需要具有改进的处理吞吐量和更低处理延迟的迭代解码算法。由于最小和算法的固有并行性，它可以以完全并行的方式操作，促进LDPC解码器具有高达16.2Gbit/s的处理吞吐量[9]。相比之下，最先进的turbo解码器的处理吞吐量[10]被限制在2.15Gbit/s。这可能归因于对数-BCJR算法的固有串行性质，这是由其前向递归和后向递归的数据依赖性所施加的[4]。更具体地，由典型的两个卷积编码器中的每一个生成的被turbo编码的比特必须跨越多个连续的时间段被串行处理，该多个连续的时间段是实际集成的电路实施方式中的时钟循环(clock cycle)。此外，对数-BCJR算法通常交替地应用于两个卷积码，直到已经执行了足够数量的解码迭代。结果，要求数千个时间段来完成最先进的turbo解码器的迭代解码处理。

[0005] 因此，提供具有较少的数据依赖性并且使能够高度并行处理的、对数-BCJR解码器的替代方案呈现技术问题。

发明内容

[0006] 根据本技术的第一示例实施例，提供了一种turbo解码器电路，用于执行turbo解码处理以从接收到的信号中恢复数据符号的帧，该数据符号的帧包括该帧的每个数据符号

的奇偶校验或奇偶校验以及系统软判决值(LLR值)。该帧的数据符号可以已经用turbo编码器使用系统码或非系统码来编码,使得接收到的该帧的软判决值可以包括系统码示例的系统和奇偶校验符号或非系统码的奇偶校验符号的软判决值。turbo解码器电路恢复已经用turbo编码器编码的帧的数据符号,该turbo编码器包括每个能够由网格表示的上部卷积编码器和下部卷积编码器、以及在上部卷积编码器和下部卷积编码器之间交织编码数据的交织器。turbo解码器电路包括时钟、用于交织软判决值的可配置网络电路、上部解码器和下部解码器。上部和下部解码器中的每一个包括处理元件,该处理元件被配置为在一系列连续时钟循环期间,从可配置网络电路迭代地接收与数据符号有关的先验软判决值(先验LLR),该数据符号与表示上部或下部卷积编码器的状态之间的可能路径的整数个连续网格级(stage)的窗口相关联。处理元件使用先验软判决值执行与该窗口相关联的并行计算,以生成与数据符号有关的对应非本征软判决值。可配置网络电路包括网络控制器电路,该网络控制器电路在连续时钟循环期间迭代地控制可配置网络电路的配置,以通过交织由下部解码器提供的非本征软判决值来为上部解码器提供先验软判决值,并且通过交织由上部解码器提供的非本征软判决值来为下部解码器提供先验软判决值。由网络控制器控制的可配置网络电路执行的交织是根据预定调度的,这在一个或多个连续时钟循环的不同循环提供先验软判决值,以避免在相同时钟循环期间向上部或下部解码器的相同处理元件提供不同先验软判决值之间的竞争。

[0007] 因此,根据本技术的示例实施例,上部解码器和下部解码器的每个处理元件执行与其网格的窗口相关联的计算。这意味着每个处理元件正在对与帧的数据符号的一部分(section)相关联且相对应的网格的一部分(section)执行与turbo解码的前向递归和后向递归相关联的计算。作为turbo解码器的任意并行处理的结果,处理元件可以划分上部解码器的网格,而不限制窗口大小到处理元件的映射,尽管可以通过在可用的处理元件之间尽可能多地共享网格级(trellis stage)的窗口大小实现更大的解码率。这也意味着帧的大小可以独立于可用于执行turbo解码的处理元件的数量而变化,从而可以动态配置通过划分网格形成的窗口大小。turbo解码电路的这种任意并行性质至少部分地是由于预定调度而实现的,该预定调度配置可配置网络,不仅根据在编码器处执行的交织来交织软判决值,而且还管理软判决值的传送,以避免在相同时钟循环中向相同处理元件传送不同软判决值而引起的竞争。

[0008] 本公开的各种其他方面和特征在所附权利要求中限定,并且包括turbo解码的方法、通信设备和无线通信网络的基础设施装备。

附图说明

[0009] 现在将仅参考附图通过示例的方式描述本公开的实施例,在附图中相同的部分被提供有对应的参考编号,并且其中:

[0010] 图1是根据LTE标准操作的移动通信系统的示意图;

[0011] 图2是图1中所示的LTE系统的示例发射器的示意框图;

[0012] 图3是图1中所示的LTE系统的示例接收器的示意框图;

[0013] 图4是简化的turbo编码器的示意框图;

[0014] 图5是示出LTE turbo编码器的更详细示例的示意框图;

[0015] 图6是表示使用形成图5的turbo编码器的一部分的卷积编码器编码的状态和状态转换的图示；

[0016] 图7是根据对数-BCJR算法的示例turbo解码器的示意框图；

[0017] 图8a是图形地示出对于对数-BCJR turbo解码器的上部和下部解码器每个中的单个处理元件的后向和前向递归的调度的示意表示，并且图8b是对数似然比的生成的对应的图；

[0018] 图9是完全并行turbo解码器的示意框图；

[0019] 图10是根据本技术的示例实施例的任意并行turbo解码器的示意框图；

[0020] 图11是根据本技术实施例的上部或下部解码器的处理元件中的一个的示例的示意框图；

[0021] 图12a、12b、12c、12d是图形地示出在处理元件(称为子处理元件)内执行的后向和前向递归的调度以生成对于在包括第二前向子处理元件的任意并行turbo解码器的上部和下部解码器每个中的单个处理元件的对数似然比的示意表示，该第二前向子处理元件操作相对于第一前向子处理元件延迟的一个时钟循环；

[0022] 图13a是被配置为计算后向子处理元件中的后向状态度量向量的、图11中所示的处理元件的一部分的示意电路图，并且图13b是被配置为计算第一前向子处理元件中的前向状态度量向量的、图11中所示的处理元件的一部分的示意框图；

[0023] 图14a是被配置为计算第一前向子处理元件中的比特度量向量的、图11中所示的处理元件的一部分的示意电路图，并且图13b是被配置为计算第二前向子处理元件中的非本征(extrinsic)和后验(posteriori)对数似然比的、图11中所示的处理元件的一部分的示意框图；

[0024] 图15是误码率相对于信噪比的曲线图，其示出了根据本技术实施例使用128个处理元件和512个符号的帧长度对于不同数量的时钟循环的任意并行turbo解码器的性能；以及

[0025] 图16是误码率相对于信噪比的曲线图，其示出了根据本技术实施例使用128个处理元件和6144个符号的帧长度、对于不同数量的时钟循环的任意并行turbo解码器的性能。

具体实施方式

[0026] 示例通信系统

[0027] 图1提供了传统移动电信系统的示意图，其中该系统包括移动通信设备104、基础设施装备101和核心网络102。基础设施装备也可以被称为例如基站、网络元件、增强型节点B(enhanced Node B，eNodeB)或协调实体，并且向覆盖区域或小区内的一个或多个通信设备提供无线接入接口。一个或多个移动通信设备可以经由使用无线接入接口发送和接收表示数据的信号来传输数据。网络实体101可通信地链接到核心网络102，其中核心网络可以连接到具有与由通信设备104和基础设施装备102形成的结构类似的结构的一个或多个其他通信系统或网络。核心网络还可以为由网络实体所服务的通信设备提供包括认证、移动性管理、计费等的功能。图1的移动通信设备也可以被称为通信终端、用户设备(user equipment，UE)、终端设备等，并且被配置为经由网络实体与由相同或不同覆盖区域服务的一个或多个其他通信设备通信。通信系统可以根据任何已知的协议来操作，例如，在一些示

例中，系统可以根据3GPP长期演进(LTE)标准来操作，其中网络实体和通信设备一般地分别被称为eNodeB和UE。

[0028] 如将从上面解释的操作中理解，UE和eNodeB的物理层被配置为发送和接收表示数据的信号。因此，典型的发射器/接收器链如图2和3所示。

[0029] 图2提供了示出了组成发射器的组件的示意框图，该发射器可以经由如图1中所示的LTE系统的无线接入接口形成物理层传输的e-NodeB 101或通信设备104的一部分。在图2中，数据经由数据格式化器204处的输入接收，并形成用于传输的帧或子帧。然后，数据帧被纠错编码器206利用纠错码来进行编码，并被馈送到符号形成器208，该符号形成器208将纠错编码的比特形成为比特组，以用于映射到用于调制的符号上。然后，数据符号被符号交织器210交织，并被馈送到OFDM调制器212，该OFDM调制器212用已经从交织器210接收的数据符号来调制OFDM符号的子载波。然后，OFDM符号被转换成RF频率，并由发射器214经由天线216发送。

[0030] 相对应地，操作以经由LTE无线接入接口接收经由物理层为通信设备104或eNodeB 101发送的数据的接收器包括接收器天线301，该接收器天线301检测经由无线接入接口发送到射频接收器302的射频信号。图3表示接收器的简化版本，并且几个块将组成OFDM解调器/均衡器304，该OFDM解调器/均衡器304将时域OFDM符号转换到频域，并解调OFDM符号的子载波以恢复数据符号并执行解交织(deinterleave)等。然而，OFDM解调器/均衡器304的输出是将表示数据比特的编码的软判决值馈送到turbo解码器306。turbo解码器执行turbo解码算法，以检测和恢复发送的数据比特的估计，该发送的数据比特被输出为与发射器的输入相对应的输出308上的数据比特流。

[0031] 对于如上所述的LTE的示例，图4示出了图2中所示的纠错编码器206的示例实施例。图4提供了示出简化的turbo编码器的示例表示，该简化的turbo编码器对包括K_l个比特的消息帧$\mathbf{b}_1^{\mathrm{u}}=\left[b_{1,k}^{\mathrm{u}}\right]_{k=1}^{K_l}$进行编码，其中每个比特具有二进制值$b_{1,k}^{\mathrm{u}}\in\{0,1\}$。如图4所示，该消息帧被提供给上部卷积编码器401和下部卷积编码器403。上部卷积编码器401执行卷积编码处理，诸如下面提供的示例，以生成两个K_l比特的编码帧，即奇偶校验帧$\mathbf{b}_2^{\mathrm{u}}=\left[b_{2,k}^{\mathrm{u}}\right]_{k=1}^{K_l}$和系统帧$\mathbf{b}_3^{\mathrm{u}}=\left[b_{3,k}^{\mathrm{u}}\right]_{k=1}^{K_l}$。同时，消息帧由内部turbo编码交织器404交织，以便获得K_l比特交织的消息帧$\mathbf{b}_1^{\mathrm{l}}=\left[b_{1,k}^{\mathrm{l}}\right]_{k=1}^{K_l}$，如图4所示，该消息帧被提供给下部卷积编码器403，该下部卷积编码器403也应用卷积编码器来生成再两个K_l比特的编码帧，即奇偶校验帧$\mathbf{b}_2^{\mathrm{l}}=\left[b_{2,k}^{\mathrm{l}}\right]_{k=1}^{K_l}$和系统帧$\mathbf{b}_3^{\mathrm{l}}=\left[b_{3,k}^{\mathrm{l}}\right]_{k=1}^{K_l}$，尽管后者没有被发送。这里，上标“u”和“l”分别指示与上部卷积编码器401和下部卷积编码器403的相关性。然而，在下文中，这些上标仅在必要时用于明确区分turbo编码器的两个卷积编码器401、403，并且当讨论同样应用于两者时被省略。注意，turbo编码器通过发送三个编码帧来表示K_l个比特的消息帧$\mathbf{b}_1^{\mathrm{u}}$，该三个编码帧总共包括$3K_l$比特，并得到$R=K_l/(3K_l)=1/3$的turbo编码率。

[0032] 如以上参考图2所解释的，在turbo编码之后，编码帧可以被调制到无线信道上并被发送到接收器，例如图3中提供的示例。

[0033] LTE turbo编码器

[0034] 图5中提供了LTE turbo编码器[1]的更具体的图示，其还示出了终止机制。对于图5中所示的示例，turbo编码器是1/3速率编码，其中从如图2中所示的数据格式化器204接收的数据比特被馈送到上部卷积编码处理器401。从图5中可以看出，所接收的K_l比特的消息帧$\mathbf{b}_1^{\mathrm{u}} = \left[b_{1,k}^{\mathrm{u}}\right]_{k=1}^{K_l}$也经由turbo码内部交织器404被馈送到下部卷积编码处理器403。根据已知的布置，K_l比特的消息帧$\mathbf{b}_1^{\mathrm{u}} = \left[b_{1,k}^{\mathrm{u}}\right]_{k=1}^{K_l}$被馈送到连接到其他存储器元件406的存储器元件406，以形成移位寄存器类型的布置。存储器元件406的输出用于形成到XOR单元408的输入，该XOR单元408在其输出处从其输入的逻辑XOR形成比特，这形成了编码的输出比特或被反馈给存储器元件406之一作为输入的比特。上部卷积编码器中的开关410在上部卷积编码器414的输入412和输出之间切换输入比特，以分别在第一输出416上形成系统帧$\mathbf{b}_3^{\mathrm{u}} = \left[b_{3,k}^{\mathrm{u}}\right]_{k=1}^{K_l}$，并在第三输出426上形成三个消息终止比特$\left[b_{3,k}^{\mathrm{u}}\right]_{k=K_l+1}^{K_l+3}$。上部卷积编码器401的第二输出418提供奇偶校验帧$\mathbf{b}_2^{\mathrm{u}} = \left[b_{2,k}^{\mathrm{u}}\right]_{k=1}^{K_l+3}$。在图5中，三个消息终止比特$\left[b_{3,k}^{\mathrm{u}}\right]_{k=K_l+1}^{K_l+3}$用于在已知状态下终止上部卷积编码器401，为简单起见，这在图4中未示出。

[0035] 在下部卷积编码器403中，开关420在从内部交织器404接收的比特之间切换，并且对应于上部卷积编码器的开关410。以类似于上部卷积编码器的方式，下部卷积编码器的输出信道422、424分别提供奇偶校验帧$\mathbf{b}_2^{\mathrm{l}} = \left[b_{2,k}^{\mathrm{l}}\right]_{k=1}^{K_l+3}$和三个消息终止比特$\left[b_{3,k}^{\mathrm{l}}\right]_{k=K_l+1}^{K_l+3}$。下部卷积编码器$\mathbf{b}_3^{\mathrm{l}} = \left[b_{3,k}^{\mathrm{l}}\right]_{k=1}^{K_l}$的系统数据比特不是从下部卷积编码器输出的，因为它们已经存在于第一输出416上。因此，在第一输出416提供输入比特作为系统码、并且第二输出418和第四输出422提供相应的奇偶校验比特的情况下，turbo编码器提供1/3速率编码。与上部卷积编码器一样，三个消息终止比特$\left[b_{3,k}^{\mathrm{l}}\right]_{k=K_l+1}^{K_l+3}$用于在已知状态下终止下部卷积编码器403，为简单起见，这在图4中未示出。

[0036] 总之，图5的LTE turbo编码器[1]采用12个附加终止比特来迫使每个卷积编码器进入最终状态$S_{K_l+3} = 1$。更具体地，上部卷积编码器401生成三个消息终止比特b_{3,K_l+1}^{u}、b_{3,K_l+2}^{u}、b_{3,K_l+3}^{u}，以及三个奇偶校验终止比特帧b_{2,K_l+1}^{u}、b_{2,K_l+2}^{u}、b_{2,K_l+3}^{u}。下部卷积编码器403以类似的方式操作，生成对应的三个消息终止比特b_{3,K_l+1}^{l}、b_{3,K_l+2}^{l}、b_{3,K_l+3}^{l}以及三个奇偶校验终止比特b_{2,K_l+1}^{l}、b_{2,K_l+2}^{l}、b_{2,K_l+3}^{l}的集合。与由上部卷积编码器产生的系统帧$\mathbf{b}_3^{\mathrm{u}}$相反，下部卷积编码器的系统帧$\mathbf{b}_3^{\mathrm{l}}$不由LTE turbo编码器输出。因此，LTE turbo编码器使用总共$(3K_l+12)$

个比特来表示K_1比特的消息帧$\mathbf{b}_1^u$，给出了更精确的编码率$R=K_1/(3K_1+12)$。

[0037] 图5中呈现的turbo编码器的示例提供了上部卷积编码器401和下部卷积编码器403，其每个具有三个存储器元件406。如熟悉卷积编码器的人已知的，存储器元件406的二进制内容可以被解释为状态，使得卷积编码处理可以被合成为通过包括卷积编码器的可能状态的网格的转换。这样，卷积编码器或turbo编码器可以被描述为马尔可夫处理，并且因此被表示为网格图。用于卷积编码器的状态转换图的示例如图6中所示。图6的状态转换图表示具有$M=8$个状态和每状态的$K=2$个转换的网格的一级(one stage)，并且因此可以提供与以相同方式操作的上部卷积编码器401和下部卷积编码器403相对应的示例。对于上部卷积编码器401，通过考虑对应的消息比特$b_{1,k}$，从初始状态$S_0=1$开始，并相继转换到每个后续状态$S_k\in\{1,2,...,M\}$。因为对于消息比特$b_{1,k}\in\{0,1\}$有两个可能的值，所以对于可以通过从先前状态S_{k-1}转换来达到的状态S_k有$K=2$个可能的值。例如，在图6中，$S_{k-1}=1$的先前状态意味着后续状态选自$S_k\in\{1,5\}$。这个示例也可以用符号$c(1,1)=1$和$c(1,5)=1$来表示，其中$c(S_{k-1},S_k)=1$指示卷积编码器可以从S_{k-1}转换到S_k，而$c(S_{k-1},S_k)=0$指示这种转换是不可能的。在$K=2$选项中，选择状态S_k的值使得$b_1(S_{k-1},S_k)=b_{1,k}$。例如，在图6中，$S_{k-1}=1$和$b_{1,k}=0$给出$S_k=1$，而$S_{k-1}=1$和$b_{1,k}=1$给出$S_k=5$。接着，根据$b_{2,k}=b_2(S_{k-1},S_k)$和$b_{3,k}=b_3(S_{k-1},S_k)$，为奇偶校验帧b_2和系统帧b_3中的对应的比特选择二进制值。在图6的示例中，$S_{k-1}=1$和$S_k=1$给出$b_{2,k}=0$和$b_{3,k}=0$，而$S_{k-1}=1$和$S_k=5$给出$b_{2,k}=1$和$b_{3,k}=1$。

[0038] ——————————————————————————————————————

[0039] LTE turbo解码器的示例

[0040] 在它们通过无线信道传输之后，如图4所示由turbo编码器生成的三个编码帧$\mathbf{b}_2^u$、$\mathbf{b}_3^u$和$\mathbf{b}_2^l$可以被解调并提供给图7的turbo解码器。然而，由于无线信道中噪声的影响，解调器将无法确定这些编码帧中的比特值。因此，代替提供包括K_1个硬值(hard-valued)比特的帧，解调器提供三个帧，每个帧包括K_1个软值先验对数似然比(LLR)$\bar{\mathbf{b}}_2^{u,a}=[\bar{b}_{2,k}^{u,a}]_{k=1}^{K_l}$、$\bar{\mathbf{b}}_3^{u,a}=[\bar{b}_{3,k}^{u,a}]_{k=1}^{K_l}$和$\bar{\mathbf{b}}_2^{l,a}=[\bar{b}_{2,k}^{l,a}]_{k=1}^{K_l}$。此外，第四帧$\bar{\mathbf{b}}_3^{l,a}=[\bar{b}_{3,k}^{l,a}]_{k=1}^{K_l}$也可以通过交织$\bar{\mathbf{b}}_3^{u,a}$的LLR来获得。这里，与比特$b_{j,k}$有关的LLR被定义为：

[0041] $$\bar{b}_{j,k}=\ln\frac{\Pr(b_{j,k}=1)}{\Pr(b_{j,k}=0)},$$

[0042] 其中上标“a”、“e”或“p”可以被附加以分别指示先验、非本征或后验LLR。

[0043] 对数-BCJR算法通常形成通过表示马尔可夫处理的每个状态的连接的网格(诸如卷积编码器)、来执行前向递归处理和后向递归处理的解码或检测处理。对于被turbo编码的数据，执行对数-BCJR解码处理的解码器包括上部解码器和下部解码器。上部解码器和下部解码器中的每一个都各自执行前向递归处理和后向递归处理，并且为每次迭代生成馈送到上部解码器和下部解码器中的另一个的非本征LLR。

[0044] 图7提供了示出对应于图4的简化turbo编码器的、用于对数-BCJR算法的简化turbo解码器的示例实施方式的示意框图。对数-BCJR turbo解码器迭代地操作，其中I次迭代中的每一次迭代包括所示的所有处理元件或算法块的操作。

[0045] 本技术的实施例可以提供与传统算法相比具有改进的解码率的任意并行turbo解

码器。此外，与诸如在我们共同未决的国际专利申请PCT/EP2015/067527[26]中公开的完全并行turbo解码器相比，应用turbo解码的并行处理的程度可以根据可用的一个或多个处理元件的数量而不是描述编码器的网格中的级的数量来设置。

[0046] 根据用于解码由图5的编码器编码的数据帧的一个示例实施方式的LTE turbo解码器包括上部解码器和下部解码器、以及上部终止元件(TE)和下部终止元件、以及CRC单元。上部解码器使用交织器连接到下部解码器，而下部解码器使用解交织器连接到上部解码器。LTE turbo解码器通常一次解码一帧比特，并且通常支持所有L＝188帧长度{K_1,K_2,K_3,...,K_{188}}＝{40,48,56,...,6144}和LTE标准[1]的对应的交织器设计。为了发起长度为长度K_l的帧的解码，其中l∈[1,L]，上部解码器被提供有K_l个奇偶校验LLR$[b_{2,k}^{u,a}]_{k=1}^{K_l}$和$K_l$个系统LLR$[b_{3,k}^{u,a}]_{k=1}^{K_l}$。同时，下部解码器被提供有$K_l$个奇偶校验LLR$[b_{2,k}^{l,a}]_{k=1}^{K_l}$。同样，上部终止元件被提供有六个终止LLR$[b_{2,k}^{u,a}]_{k=K_l+1}^{K_l+3}$和$[b_{3,k}^{u,a}]_{k=K_l+1}^{K_l+3}$，而下部终止元件被提供有再六个终止LLR$[b_{2,k}^{l,a}]_{k=K_l+1}^{K_l+3}$和$[b_{3,k}^{l,a}]_{k=K_l+1}^{K_l+3}$。为简单起见，图7中未示出终止元件。图7示出了根据我们共同未决的国际专利申请PCT/EP2015/067527[26]中公开的BCJR算法的传统turbo解码器的示例。这将仅在以下段落中简要描述。

[0047] 如图7所示，第一组2K_l个算法块601专用于对由上部卷积编码器401产生的被turbo编码的数据执行turbo解码算法的第一部分。上部解码器601的第一行K_l个算法块610专用于通过可能状态的网格执行前向递归处理，而第二行K_l个算法块612专用于根据对数-BCJR算法通过网格级执行后向递归。每个算法块与网格中的K_l个级中的一个级相对应，其包括一组先前状态和一组下一状态之间的一组转换。第二组2K_l个算法块602专用于对由下部卷积编码器403产生的被turbo编码的数据执行turbo解码算法的第二部分。至于上部解码器601，下部解码器包括下部解码器602的第一行K_l个算法块620，该下部解码器602的第一行K_l个算法块620专用于通过可能状态的网格执行前向递归处理，而第二行K_l个算法块622专用于根据对数-BCJR算法通过网格状态执行后向递归。

[0048] 专用于执行对数-BCJR算法610的前向递归部分的、上部解码器601的K_l个算法块610中的第k个算法块610被布置成从解调器接收第k个LLR值$\bar{b}_{2,k}^{u,a}$，$\bar{b}_{3,k}^{u,a}$，该第k个LLR值是针对由上部编码器401生成的编码比特$\mathbf{b}_2^u$、$\mathbf{b}_3^u$的帧而估计的。相对应地，下部解码器602的K_l个算法块620的第k个算法块620被布置成从解调器接收第k个LLR值$\bar{b}_{2,k}^{l,a}$，$\bar{b}_{3,k}^{l,a}$，该第k个LLR值是针对由下部编码器402生成的编码比特$\mathbf{b}_2^l$、$\mathbf{b}_3^l$的帧而估计的。这里，解调器可以通过交织$\mathbf{b}_3^u$获得$\mathbf{b}_3^l$。

[0049] 每个依次被布置成在上部解码器601和下部解码器602中一个接一个地执行前向递归的第k个算法块610、620组合L＝3先验LLR$\bar{b}_{1,k}^{a}$，$\bar{b}_{2,k}^{a}$和$\bar{b}_{3,k}^{a}$，以便获得对于状态转换图(例如，如图6所示)中每个转换的先验度量$\bar{\gamma}_k(S_{k-1},S_k)$。在该计算之后，执行前向递归的第k个算法块610、620中的每一个将这些先验转换度量与先验前向状态度量$\bar{\alpha}_{k-1}(S_{k-1})$组

合，以便获得非本征前向状态度量$\bar{\alpha}_k(S_k)$。然后，这些非本征状态度量被传递到第k+1个算法块610、620，以在下一个时间段中被用作先验状态度量。然而，如熟悉对数-BCJR算法的人将理解，turbo解码器的上部解码器和下部解码器交替工作，使得当一个激活时，另一个空闲。

[0050] 在上部解码器601和下部解码器602中执行后向递归的第k个算法块612、622组合对于每个转换的先验度量$\bar{\gamma}_k(S_{k-1},S_k)$与先验后向状态度量$\bar{\beta}_k(S_k)$。这产生了非本征后向状态度量$\bar{\beta}_{k-1}(S_{k-1})$，其可以被传递到第k-1个算法块，以在下一个时间段中被用作先验状态度量。此外，第k个算法块612、622在上部解码器601和下部解码器602中执行后向递归，以获得对于状态转换图(例如，如图6所示)中每个转换的后验度量$\bar{\delta}_k(S_{k-1},S_k)$。最后，在上部解码器401和下部解码器中执行后向递归的第k个算法块612、622组合对于状态转换图中转换的后验度量$\bar{\delta}_k(S_{k-1},S_k)$，以生成第k个比特的非本征消息LLR。这些LLR值在上部解码器601和下部解码器602之间交换(swap)。

[0051] 上部解码器601和下部解码器602为帧的每个数据比特交换非本征LLR，该非本征LLR成为编码数据帧的系统比特的估计。更具体地，交织器604根据由turbo编码器的上部卷积编码器401和下部卷积编码器402所使用的数据比特的交织，对在上部解码器601和下部解码器602之间传递的数据比特的LLR值进行交织。此外，交织器604对在下部解码器602和上部解码器601之间传递的数据比特的LLR值执行解交织，以反转由turbo编码器的上部卷积编码器401和下部卷积编码器402所使用的数据比特的交织。

[0052] 如将从以上描述中理解，对于被turbo编码的数据的turbo解码通常包括贯穿解码处理操作的上部解码器和下部解码器。更具体地，上部解码器的操作更新K_1个后验LLR$[b_{1,k}^{\mathrm{u,p}}]_{k=1}^{K_l}$和$K_1$个非本征LLR$[b_{1,k}^{\mathrm{u,e}}]_{k=1}^{K_l}$的值。对非本征LLR的这些更新被交织，并作为$K_1$个先验LLR$[b_{1,k}^{\mathrm{l,a}}]_{k=1}^{K_l}$被提供给下部解码器。更具体地，$b_{1,\boldsymbol{\Pi}(k)}^{\mathrm{l,a}}=b_{1,k}^{\mathrm{u,e}}$，其中交织动作由向量Π描述，其中$K_1$个元件Π(k)∈[1,$K_1$]中的每一个是唯一的。例如，$K_1$=40比特LTE交织器可以由向量Π=[1,38,15,12,29,26,3,40,17,14,31,28,5,2,19,16,33,30,7,4,21,18,35,32,9,6,23,20,37,34,11,8,25,22,39,36,13,10,27,24]来描述，其中$b_{1,24}^{\mathrm{l,a}}=b_{1,40}^{\mathrm{u,e}}$。同样，下部解码器的操作更新$K_1$个非本征LLR$[b_{1,k}^{\mathrm{l,e}}]_{k=1}^{K_l}$的值，这些值被解交织并作为$K_1$个先验LLR$[b_{1,k}^{\mathrm{u,a}}]_{k=1}^{K_l}$被提供给上部解码器。更具体地，$b_{1,\boldsymbol{\Pi}^{-1}(k)}^{\mathrm{u,a}}=b_{1,k}^{\mathrm{l,e}}$，其中解交织动作由向量$\Pi^{-1}$描述，其中$K_1$个元件$\Pi^{-1}$(k)∈[1,$K_1$]中的每一个是唯一的，并且服从与交织器$\Pi^{-1}$(Π(k))=k的关系。例如，$K_1$=40比特LTE解交织器可以由向量$\Pi^{-1}$=[1,14,7,20,13,26,19,32,25,38,31,4,37,10,3,16,9,22,15,28,21,34,27,40,33,6,39,12,5,18,11,24,17,30,23,36,29,2,35,8]来描述，其中$b_{1,40}^{\mathrm{u,a}}=b_{1,24}^{\mathrm{l,e}}$。此外，对后验LLR$[b_{1,k}^{\mathrm{u,p}}]_{k=1}^{K_l}$的更新通常被提供给CRC单元。当使用硬判决$b_{1,k}^{\mathrm{u,p}}>0$获得的解码的比特满足LTE CRC时，这停止解码处理。然后，解码的比特和后验LLR由turbo解码器输出，并且下一帧的解码可以开始。

[0053] 使用网格级的窗口的Turbo解码的适应

[0054] 在操作以执行对数-BCJR算法的turbo解码器的一些实施方式中，奇偶校验、系统、先验、非本征和后验LLR可以在解码期间被组合成连续窗口的链，每个窗口包括相等数量W_l的每种类型的LLR。此外，turbo解码处理通常根据具有C_l个时钟循环的周期的周期性调度来完成。通常，W_l和C_l的值取决于当前帧长度K_l。然而，不同的turbo解码器采用不同的开窗(windowing)和不同的调度技术，这将在下面的部分中讨论。

[0055] 第一示例开窗

[0056] 在第一示例[21]中，LLR被分组为P＝8个连续窗口的链，因为这是LTE帧长度K_l的所有L＝188个支持值的最大公约数。这确保了所有窗口包括相同数量$W_l=K_l/P$的每种类型的LLR，而不管当前帧的长度K_l。因此，[21]的设计采用P＝8个处理元件的链，其中每个处理元件对上部解码器的窗口中的不同的一个窗口以及对下部解码器的对应窗口执行处理。因此，每个窗口执行对数-BCJR算法的计算610、612、620、622的处理。这样，具有索引$k\in[1,K_l]$的LLR由具有索引$p=\lceil k/W_l\rceil$的处理元件来处理。注意，最大窗口长度由$W_{\max}=\max_{l=1}^{L}W_l=K_{\max}/P=768$给出，其中$K_{\max}=\max_{l=1}^{L}K_l=6144$是最长LTE帧长度中的比特数。根据$C_l=2W_l$，解码处理根据周期性调度来完成，其中周期$C_l$取决于当前帧长度$K_l$。随着解码处理的进行，计数器c重复计数到$C_l$。

[0057] 具有索引$p\in[1,P]$的处理元件基于包括W_l个奇偶校验、系统、先验、非本征和后验LLR的窗口来操作。这里，符号$k'\in[1,W_l]$用于索引第p个窗口内的LLR，该符号可以根据$k=k'+(p-1)W_l$被转换为帧内的索引$k\in[1,K_l]$。在解码处理开始时，具有索引p的处理元件被提供有W_l个上部奇偶校验LLR$[b_{2,k'}^{\mathrm{u,a}}]_{k'=1}^{W_l}$、$W_l$个下部奇偶校验LLR$[b_{2,k'}^{\mathrm{l,a}}]_{k'=1}^{W_l}$和$W_l$个上部系统LLR$[b_{3,k'}^{\mathrm{u,a}}]_{k'=1}^{W_l}$。贯穿解码处理，第p个处理元件被连续地提供有对$W_l$个上部先验LLR$[b_{1,k'}^{\mathrm{u,a}}]_{k'=1}^{W_l}$和$W_l$个下部先验LLR$[b_{1,k'}^{\mathrm{l,a}}]_{k'=1}^{W_l}$的更新。作为响应，如下所述，该处理元件连续地更新$W_l$个上部非本征LLR$[b_{1,k'}^{\mathrm{u,e}}]_{k'=1}^{W_l}$、$W_l$个下部非本征LLR$[b_{1,k'}^{\mathrm{l,e}}]_{k'=1}^{W_l}$和$W_l$个上部后验LLR$[b_{1,k'}^{\mathrm{u,p}}]_{k'=1}^{W_l}$。

[0058] 每个处理元件根据下面的等式(1)至(4)对每个窗口执行计算。注意，与上部解码器不同，下部解码器不受益于系统LLR，这相当于使得$b_{3,k}^{\mathrm{a}}=0$。这允许在下部解码器的情况下，从(1)-(3)中省略对应的术语。同样，下部解码器不生成后验LLR，允许(4)被完全省略。

[0059]

$$\beta_{k-1}(s_{k-1})=\max_{\{s_k|c(s_{k-1},s_k)=1\}}\left[b_1(s_{k-1},s_k)\cdot b_{1,k}^{\mathrm{a}}+b_2(s_{k-1},s_k)\cdot b_{2,k}^{\mathrm{a}}+b_3(s_{k-1},s_k)\cdot b_{3,k}^{\mathrm{a}}+\beta_k(s_k)\right] \quad (1)$$

[0060]

$$\alpha_k(s_k)=\max_{\{s_{k-1}|c(s_{k-1},s_k)=1\}}\left[b_1(s_{k-1},s_k)\cdot b_{1,k}^{\mathrm{a}}+b_2(s_{k-1},s_k)\cdot b_{2,k}^{\mathrm{a}}+b_3(s_{k-1},s_k)\cdot b_{3,k}^{\mathrm{a}}+\alpha_{k-1}(s_{k-1})\right] \quad (2)$$

[0061]
$$b_{1,k}^{e}=0.75\left[\max_{\{(s_{k-1},s_k)|b_1(s_{k-1},s_k)=1\}}[\alpha_{k-1}(s_{k-1})+\beta_k(s_k)+b_2(s_{k-1},s_k)\cdot b_{2,k}^{a}]\right]$$
$$-0.75\left[\max_{\{(s_{k-1},s_k)|b_1(s_{k-1},s_k)=0\}}[\alpha_{k-1}(s_{k-1})+\beta_k(s_k)+b_2(s_{k-1},s_k)\cdot b_{2,k}^{a}]\right]+b_{3,k}^{a} \quad (3)$$

[0062]
$$b_{1,k}^{p}=0.75\left[\max_{\{(s_{k-1},s_k)|b_1(s_{k-1},s_k)=1\}}[\alpha_{k-1}(s_{k-1})+\beta_k(s_k)+b_2(s_{k-1},s_k)\cdot b_{2,k}^{a}]\right]$$

[0063]
$$-0.75\left[\max_{\{(s_{k-1},s_k)|b_1(s_{k-1},s_k)=0\}}[\alpha_{k-1}(s_{k-1})+\beta_k(s_k)+b_2(s_{k-1},s_k)\cdot b_{2,k}^{a}]\right]+b_{1,k}^{a}+b_{3,k}^{a} \quad (4)$$

[0064] 此外，贯穿解码处理，每个解码器中的第(p-1)个处理元件周期性地向第p个处理元件提供对用于k′＝0的情况的上部前向状态度量向量$\alpha_{k'}^{u}=[\alpha_{k'}^{u}(s_{k'})]_{s_{k'}=1}^{S}$和下部前向状态度量向量$\alpha_{k'}^{l}=[\alpha_{k'}^{l}(s_{k'})]_{s_{k'}=1}^{S}$的更新。但是，如果p＝1并且因此第(p-1)个处理元件不存在，则这是例外。在这些情况下，第p个处理元件采用$\alpha_0^{u}=\alpha_0^{l}=[0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty]$。同样，每个解码器中的第(p+1)个处理元件周期性地向第p个处理元件提供用于k′＝$W_{1,p}$的情况的上部后向状态度量向量$\beta_{k'}^{u}=[\beta_{k'}^{u}(s_{k'})]_{s_{k'}=1}^{S}$和下部后向状态度量向量$\beta_{k'}^{l}=[\beta_{k'}^{l}(s_{k'})]_{s_{k'}=1}^{S}$的更新。然而，如果p＝P，则这是例外，在这种情况下，后向状态度量向量由对应的终止元件提供。如下所述，第p个处理元件周期性地更新分别提供给第(p+1)和(p-1)个处理元件的、对于k′＝$W_{1,p}$的上部前向状态度量向量$\alpha_{k'}^{u}$和下部前向状态度量向量$\alpha_{k'}^{l}$、以及对于k′＝0的上部后向状态度量向量$\beta_{k'}^{u}$和下部后向状态度量向量$\beta_{k'}^{l}$。

[0065] 处理元件根据图8中所示的调度进行操作。图8提供了图形表示，示出了处理元件如何调度(a)(1)和(2)的后向和前向递归、以及(b)对于第一开窗示例的turbo解码器中的单个处理元件的(3)和(4)的LLR的生成。处理元件对上部解码器的窗口执行(1)到(4)，接着对下部解码器的相对应的窗口执行(1)到(3)。这里，执行后向递归，其中递减的LLR索引k′在连续的时钟循环中被处理。同时，执行前向递归，其中递增的LLR索引k′在连续的时钟循环中被处理。在C_1个时钟循环的每个周期中的前$C_1/2$个时钟循环期间，处理元件执行上部解码器的前向递归和后向递归。此后，在C_1个时钟循环的每个周期中的后$C_1/2$个时钟循环期间，执行下部解码器的前向递归和后向递归，如图8所示。注意，以下讨论同样适用于上部和下部解码器，因此上标“u”和“l”从符号中移除。

[0066] 后向递归基于(1)操作，以便生成后向状态度量向量$\beta_{k'-1}$。这被存储在内部存储器中，以便促进其在递归的下一步中用作$\beta_{k'}$，以及用于非本征LLR $b_{1,k'}^{e}$和后验LLR $b_{1,k'}^{p}$的计算，如下所述。此外，在k′＝1的情况中，后向状态度量向量$\beta_{k'-1}$被存储在输出寄存器中，以便提供β_0以初始化相邻处理元件中的后向递归。同样，前向递归基于(2)操作，以便生成前向状态度量向量$\alpha_{k'}$。这被存储在内部存储器中，以便促进其在递归的下一步中用作$\alpha_{k'-1}$、以及用于非本征LLR $b_{1,k'}^{e}$和后验LLR $b_{1,k'}^{p}$的计算，如下所述。此外，在k′＝W_1的情况下，前向状态度量

向量$\alpha_{k'}$被存储在输出寄存器中，以便提供α_{W_l}以初始化相邻处理元件中的前向递归。

[0067] 如图8中所示，在递归的后半部分的每个时钟循环中，由每个处理元件输出两个后验LLR和两个非本征LLR。更具体地，在为上部解码器执行的每个递归的后半部分的每个时钟循环期间，(4)用于生成后验LLR$b_{1,k'}^{u,p}$，该后验LLR$b_{1,k'}^{u,p}$由处理元件输出并提供给CRC单元。此外，在对上部和下部解码器两者执行的每个递归的后半部分的每个时钟循环期间，(3)被用于生成非本征LLR$b_{1,k'}^{e}$，该非本征LLR$b_{1,k'}^{e}$由处理元件输出并被适当地提供给交织器或解交织器。交织器和解交织器将非本征LLR传送到适当的处理元件，其中它们被存储在存储器中，准备如上所述被用作先验LLR。注意，当P是K_1的整数除数时，LTE交织器被专门设计为无竞争，这适用于P＝8时的K_1的所有188个可能值。由于这个原因，在递归的后半部分的每个时钟循环中，正好有两个LLR被传送到每个处理元件。

[0068] 第二示例开窗

[0069] 从[22]提供的第二示例以与第一示例的方式类似的方式操作，但是利用了所支持的LTE帧长度的不同连续子集具有不同的最大公约数8、16、32和64的观察。更具体地，取决于帧长度K_1，该设计采用8、16、32或64个窗口，每个窗口包括相同数量W_1的每种类型的LLR，根据

[0070]
$$W_l=\begin{cases}K_l/8 & \text{如果 } K_l\in\{40,48,56,K,504\}\\ K_l/16 & \text{如果 } K_l\in\{512,528,544,K,1008\}\\ K_l/32 & \text{如果 } K_l\in\{1024,1058,1088,K,2016\}\\ K_l/64 & \text{如果 } K_l\in\{2048,2112,2176,K,6144\}\end{cases} \quad (5)$$

[0071] 这种设计采用了P＝64个处理元件的链，尽管这些处理元件中的一些在解码较短的帧长度时被去激活。更具体地，被激活的处理元件的数量等于所使用的窗口的数量，使得每个处理元件可以对上部解码器的窗口中的不同的一个窗口以及对于下部解码器的对应窗口执行处理。

[0072] 处理元件对每个窗口的处理根据(6)-(9)完成。注意，与上部解码器不同，下部解码器不受益于系统LLR，这相当于使得$b_{3,k}^{a}=0$。这允许在下部解码器的情况下，从(6)-(8)中省略对应的术语。同样，下部解码器不生成后验LLR，允许(9)被完全省略。

[0073]
$$\beta_{k-1}(s_{k-1})=\max_{\{s_k|c(s_{k-1},s_k)=1\}}\left[b_1(s_{k-1},s_k)\cdot b_{1,k}^{a}+b_2(s_{k-1},s_k)\cdot b_{2,k}^{a}+b_3(s_{k-1},s_k)\cdot b_{3,k}^{a}+\beta_k(s_k)\right]-\max_{s_{k-1}=1}^{S}\left[\max_{\{s_k|c(s_{k-1},s_k)=1\}}\left[b_1(s_{k-1},s_k)\cdot b_{1,k}^{a}+b_2(s_{k-1},s_k)\cdot b_{2,k}^{a}+b_3(s_{k-1},s_k)\cdot b_{3,k}^{a}+\beta_k(s_k)\right]\right] \quad (6)$$

[0074]

$$\alpha_k(s_k) = \max_{\{s_{k-1}|c(s_{k-1},s_k)=1\}}\left[b_1(s_{k-1},s_k)\cdot b^{a}_{1,k} + b_2(s_{k-1},s_k)\cdot b^{a}_{2,k} + b_3(s_{k-1},s_k)\cdot b^{a}_{3,k} + \alpha_{k-1}(s_{k-1})\right] - \max_{s_k=1}^{S}\left[\max_{\{s_{k-1}|c(s_{k-1},s_k)=1\}}\left[b_1(s_{k-1},s_k)\cdot b^{a}_{1,k} + b_2(s_{k-1},s_k)\cdot b^{a}_{2,k} + b_3(s_{k-1},s_k)\cdot b^{a}_{3,k} + \alpha_{k-1}(s_{k-1})\right]\right]$$

（7）

[0075]

$$b^{e}_{1,k} = 0.75\left[\max_{\{(s_{k-1},s_k)|b_1(s_{k-1},s_k)=1\}}\left[\alpha_{k-1}(s_{k-1})+\beta_k(s_k)+b_2(s_{k-1},s_k)\cdot b^{a}_{2,k}\right]\right] - 0.75\left[\max_{\{(s_{k-1},s_k)|b_1(s_{k-1},s_k)=0\}}\left[\alpha_{k-1}(s_{k-1})+\beta_k(s_k)+b_2(s_{k-1},s_k)\cdot b^{a}_{2,k}\right]\right] + b^{a}_{3,k} \quad (8)$$

[0076]

$$b^{p}_{1,k} = 0.75\left[\max_{\{(s_{k-1},s_k)|b_1(s_{k-1},s_k)=1\}}\left[\alpha_{k-1}(s_{k-1})+\beta_k(s_k)+b_2(s_{k-1},s_k)\cdot b^{a}_{2,k}\right]\right] - 0.75\left[\max_{\{(s_{k-1},s_k)|b_1(s_{k-1},s_k)=0\}}\left[\alpha_{k-1}(s_{k-1})+\beta_k(s_k)+b_2(s_{k-1},s_k)\cdot b^{a}_{2,k}\right]\right] + b^{a}_{1,k} + b^{a}_{3,k} \quad (9)$$

[0077] 对第一示例的turbo解码器设计的另一差异在于，分别计算(6)和(7)的后向状态度量向量$\beta_{k'-1}$和前向状态度量向量$\alpha_{k'}$。在后向和前向递归的连续时钟循环中，这些状态度量可以无限制地增长，这可能导致固定点实施方式中的溢出。[23]的模归一化(module normalisation)方案利用了这样的观察，即每个向量中的状态度量的绝对值并不重要，相反，重要的是这些状态度量之间的差异。这激发了每个向量在其生成期间内归一化状态度量。为了最小化溢出的发生，归一化是通过减去每个向量中状态度量的最大值来实现的，如(6)和(7)中所示。

[0078] 第三示例开窗

[0079] 第三示例是所谓的“混洗(shuffled)”turbo解码器[24]，其以类似于示例1和2的turbo解码器的方式操作，但是采用专用于执行上部解码器的解码的P个处理元件的一个链、以及专用于执行下部解码器的解码的P个处理元件的第二链，其中P是K_1的整数除数。每个处理元件对于对应解码器中的窗口中的不同的一个窗口执行处理，其中每个窗口具有长度$W_1=K_1/P$。与第一和第二示例的turbo解码器相比，解码处理根据周期性调度完成，其中周期由$C_1=W_1$而不是由$C_1=2W_1$给出。更具体地，上部解码器的后向和前向递归与下部解码器的后向和前向递归同时执行。在调度的特定时钟循环中由一个解码器生成的非本征LLR立即通过交织器或解交织器被传递到另一个解码器，在另一个解码器中它们可以在调度的下一个时钟循环中用作先验LLR。

[0080] 第四示例完全并行turbo解码器

[0081] 完全并行turbo解码器(fully parallel turbo decoder,FPTD)，诸如在我们共同未决的国际专利申请PCT/EP2015/067527中公开的，以类似于第一示例的turbo解码器的方式操作，但是使用K_1个窗口，每个窗口的长度为$W_1=1$。因此，FPTD采用$P=K_1$个处理元件的链，其中每个处理元件对上部解码器的窗口中的不同的一个窗口以及对下部解码器的对应窗口执行处理。FPTD解码处理根据周期性调度完成，其中周期由$C_1=2$给出。每个窗口的处

理根据下面的(10)-(15)完成。注意,在下部解码器的情况下,可以完全省略(15),因为它不生成后验LLR。这里,(10)和(13)中的上标“c”表示时钟循环索引,而(11)、(12)、(14)和(15)中的上标“c-1”表示先前时钟循环的索引。包括该符号以便强调(10)的转换度量向量和(13)的比特度量向量是流水线式的。

[0082] $$\gamma_k^c(s_{k-1},s_k)=0.75\cdot b_1(s_{k-1},s_k)\cdot b_{1,k}^{\mathrm{a}}+b_2(s_{k-1},s_k)\cdot b_{2,k}^{\mathrm{a}}+b_3(s_{k-1},s_k)\cdot b_{3,k}^{\mathrm{a}} \quad (10)$$

[0083] $$\beta_{k-1}(s_{k-1})=\max_{\{s_k|c(s_{k-1},s_k)=1\}}\left[\gamma_k^{c-1}(s_{k-1},s_k)+\beta_k(s_k)\right]-\max_{\{s_k|c(1,s_k)=1\}}\left[\gamma_k^{c-1}(1,s_k)+\beta_k(s_k)\right] \quad (11)$$

[0084] $$\alpha_k(s_k)=\max_{\{s_{k-1}|c(s_{k-1},s_k)=1\}}\left[\gamma_k^{c-1}(s_{k-1},s_k)+\alpha_{k-1}(s_{k-1})\right]-\max_{\{s_{k-1}|c(s_{k-1},1)=1\}}\left[\gamma_k^{c-1}(s_{k-1},1)+\alpha_{k-1}(s_{k-1})\right] \quad (12)$$

[0085] $$\epsilon_k^c(b'_{1,k},b'_{2,k})=\left[\max_{\left\{(s_{k-1},s_k)\middle|\begin{matrix}b_1(s_{k-1},s_k)=b'_{1,k}\\ b_2(s_{k-1},s_k)=b'_{2,k}\end{matrix}\right\}}\left[\alpha_{k-1}(s_{k-1})+\beta_k(s_k)\right]\right] \quad (13)$$

[0086] $$b_{1,k}^{\mathrm{e}}=\left[\max_{b'_{2,k}\in\{0,1\}}\left[\varepsilon_k^{c-1}(1,b'_{2,k})+b'_{2,k}\cdot b_{2,k}^{\mathrm{a}}\right]\right]-\left[\max_{b'_{2,k}\in\{0,1\}}\left[\varepsilon_k^{c-1}(0,b'_{2,k})+b'_{2,k}\cdot b_{2,k}^{\mathrm{a}}\right]\right] \quad (14)$$

[0087] $$b_{1,k}^{\mathrm{p}}=\left[\max_{b'_{2,k}\in\{0,1\}}\left[\varepsilon_k^{c-1}(1,b'_{2,k})+b'_{2,k}\cdot b_{2,k}^{\mathrm{a}}\right]\right]-\left[\max_{b'_{2,k}\in\{0,1\}}\left[\varepsilon_k^{c-1}(0,b'_{2,k})+b'_{2,k}\cdot b_{2,k}^{\mathrm{a}}\right]\right]+b_{1,k}^{\mathrm{a}}+b_{3,k}^{\mathrm{a}} \quad (15)$$

[0088] 不是在每个周期的第一时钟循环中对上部解码器执行所有处理,而是在第二时钟循环中对下部解码器执行处理之前,FPTD采用由LTE交织器的奇偶性质激发的奇偶调度。此外,FPTD采用流水线技术,以便最大化可实现的时钟频率。更具体地,在每个周期的第一时钟循环期间,具有奇数索引的处理元件对上部解码器的对应窗口执行(10)、(14)和(15)的处理,以及对下部解码器的对应窗口执行(11)、(12)和(13)的处理。同时,具有偶数索引的处理元件对下部解码器的对应窗口执行(10)和(14)的处理,以及对上部解码器的对应窗口执行(11)、(12)和(13)的处理。在每个周期的第二时钟循环中,具有偶数索引的处理元件对上部解码器的对应窗口执行(10)、(14)和(15)的处理,以及对下部解码器的对应窗口执行(11)、(12)和(13)的处理。同时,具有奇数索引的处理元件对下部解码器的对应窗口执行(10)和(14)的处理,以及对上部解码器的对应窗口执行(11)、(12)和(13)的处理。注意,在用于(11)和(12)的FPTD中使用的归一化技术不同于第二示例的归一化技术。更具体地,为了移除确定每个向量中最大状态度量的要求,(11)和(12)的方案总是减去第一状态度量。还要注意,FPTD受益于向下部解码器提供系统LLR$[b_{3,k}^{1,\mathrm{a}}]_{k=1}^{K_l}$,这可以通过交织上部解码器的系统LLR$[b_{3,k}^{\mathrm{u},\mathrm{a}}]_{k=1}^{K_l}$来获得。

[0089] 图9提供了如在PCT/EP2015/067527中公开的FPTD的图示。在图9中,相应的上部

turbo解码部分701和下部turbo解码部分702对应于对数-BCJR算法601、602的上部turbo解码部分和下部turbo解码部分，但是被K_1个并行算法块706、708代替。因此，上部解码器701由K_1个算法块706组成，而下部解码器702由K_1个算法块708组成。如图9所示，并且对应于对数-BCJR算法的操作，图3的接收器中的解调器向如图9所示被布置成两行的、turbo解码器的$2K_1$个算法块708、706提供先验LLR。更具体地，在它们通过无线信道的传输之后，三个编码帧被解调并被提供给图9的turbo解码器。解调器提供三个帧，每个帧包括K_1个软值先验对数似然比(LLR) $\bar{\mathbf{b}}_2^{\mathrm{u,a}}=\left[\bar{b}_{2,k}^{\mathrm{u,a}}\right]_{k=1}^{K_l}$、$\bar{\mathbf{b}}_3^{\mathrm{u,a}}=\left[\bar{b}_{3,k}^{\mathrm{u,a}}\right]_{k=1}^{K_l}$和$\bar{\mathbf{b}}_2^{\mathrm{l,a}}=\left[\bar{b}_{2,k}^{\mathrm{l,a}}\right]_{k=1}^{K_l}$。而第四帧$\bar{\mathbf{b}}_3^{\mathrm{l,a}}=\left[\bar{b}_{3,k}^{\mathrm{l,a}}\right]_{k=1}^{K_l}$通过交织$\bar{\mathbf{b}}_3^{\mathrm{u,a}}$获得。这些被提供给完全并行turbo解码器的$2K_1$个算法块，其中先验奇偶校验LLR $\bar{b}_{2,k}^{\mathrm{u,a}}$和先验系统LLR $\bar{b}_{3,k}^{\mathrm{u,a}}$被提供给图9中所示的上部解码器701中的第k个算法块706。此外，如将在下面详细描述的，交织器704向上部解码器701中的第k个算法块提供先验消息LLR $\bar{b}_{1,k}^{\mathrm{u,a}}$。同时，下部解码器702中的第k个算法块相应地被提供有先验LLR值$\bar{b}_{1,k}^{\mathrm{l,a}}$、$\bar{b}_{2,k}^{\mathrm{l,a}}$和$\bar{b}_{3,k}^{\mathrm{l,a}}$。除此之外，如将在下面详细描述的，上部解码器701和下部解码器702的每一个中的第k个算法块706、708还被提供有先验前向状态度量的向量$\bar{\boldsymbol{\alpha}}_{k-1}=\left[\bar{\alpha}_{k-1}(S_{k-1})\right]_{S_{k-1}=1}^{M}$和先验后向状态度量的向量$\bar{\boldsymbol{\beta}}_k=\left[\bar{\beta}_k(S_k)\right]_{S_k=1}^{M}$。与根据上面参考图7描述的对数-BCJR算法操作的传统turbo解码器不同，上部解码器701和下部解码器702的算法块706、708中的每一个以相同的方式操作，以接收对应于与网格级相关联的一个或多个数据符号的上部解码器701软判决先验LLR值$\bar{\mathbf{b}}_2^{\mathrm{u,a}}=\left[\bar{b}_{2,k}^{\mathrm{u,a}}\right]_{k=1}^{K_l}$和$\bar{\mathbf{b}}_3^{\mathrm{u,a}}=\left[\bar{b}_{3,k}^{\mathrm{u,a}}\right]_{k=1}^{K_l}$或下部解码器702的软判决先验LLR值$\bar{\mathbf{b}}_2^{\mathrm{l,a}}=\left[\bar{b}_{2,k}^{\mathrm{l,a}}\right]_{k=1}^{K_l}$和$\bar{\mathbf{b}}_3^{\mathrm{l,a}}=\left[\bar{b}_{3,k}^{\mathrm{l,a}}\right]_{k=1}^{K_l}$，并且从一个相邻算法块接收先验前向状态度量$\bar{\boldsymbol{\alpha}}_{k-1}$，从第二相邻算法块接收先验后向状态度量$\bar{\boldsymbol{\beta}}_k$，并且从第二检测处理器接收针对与第k个算法块相关联的网格级检测的数据符号的先验LLR值$\bar{b}_{1,k}^{\mathrm{a}}$。每个算法块执行与一个网格级相关联的计算，包括一组先前状态和一组下一状态之间的一组转换。根据等式(10)至(15)，每个算法块被配置为组合先验前向状态度量$\bar{\boldsymbol{\alpha}}_{k-1}=\left[\bar{\alpha}_{k-1}(S_{k-1})\right]_{S_{k-1}=1}^{M}$、先验后向状态度量$\bar{\boldsymbol{\beta}}_k=\left[\bar{\beta}_k(S_k)\right]_{S_k=1}^{M}$和与数据符号相关的先验LLR值$\bar{b}_{1,k}^{\mathrm{a}}$。

[0090] 任意并行turbo解码器的示例实施例

[0091] 如以上所解释的，在PCT/EP2015/067527中公开的FPTD提供了一种用于执行turbo解码的并行处理的布置，该布置移除了处理元件之间的依赖性，这允许每个处理元件并行地执行与每个网格级相对应的计算，并具有增加的吞吐量。因此，实际上，窗口大小如以上所解释。然而，FPTD存在明显的缺点在于，使用FPTD执行turbo解码的算法和计算要求在解码器处提供LLR值的帧中的每一个符号由处理元件表示。对于LTE帧的示例，所需求的处理器的数量将是$K_{188}=6144$，因为这是所支持的最长帧长度。这可以被认为是一个缺点，因为

只有这些处理器的有限子集可以用于较短的帧长度,导致降低的硬件效用。因此,将期望找到一种布置,其中可以利用任意数量的处理元件来实现turbo解码,允许在吞吐量和要对撞的(to be struck)硬件效用之间达成期望的权衡。这种布置在以下段落中被称为任意并行turbo解码器。因此,根据本技术的实施例,任意并行turbo解码器(arbitrary parallel turbo decoder,APTD)被布置成使用任意数量的处理元件、使用并行处理来执行进一步的解码。为此,每个处理元件表示与在发送的帧中的多个符号相对应的多个LLR值的处理算法中使用的变量的计算,使得可以减少处理元件的数量。然而,为了提供任意并行turbo解码器,有必要调整上部和下部解码器之间的符号的交织,因为每个处理元件正在对多个帧长度执行计算。结果,提供了被称为Benes网络的可配置网络,该网络被调度以在上部解码器和下部解码器之间提供符号的最优切换,这尽可能地防止其中一个或多个处理元件空闲的冲突或等待循环。将在以下段落中将更好地理解本技术的实施例。本技术的实施例可以提供一种turbo解码器电路,用于执行turbo解码处理以从接收到的信号中恢复数据符号的帧,该数据符号的帧包括该帧的每个数据符号的奇偶校验和系统软判决值(LLR值),例如其中该帧的数据符号已经用turbo编码器使用帧的每个数据符号的系统码或奇偶校验软判决值来编码,例如其中该帧的数据符号已经用turbo编码器使用非系统码来编码。因此,由接收到的信号表示的帧可以已经用系统或非系统码进行了编码。turbo解码器电路恢复已经用turbo编码器编码的帧的数据符号,该turbo编码器包括每个能够由网格表示的上部卷积编码器和下部卷积编码器、以及在上部卷积编码器和下部卷积编码器之间交织编码数据的交织器。turbo解码器电路包括时钟、被配置为交织软判决值的可配置网络电路、上部解码器和下部解码器。

[0092] 上部解码器包括与上部卷积编码器相关联的多个上部处理元件,上部解码器的处理元件中的每一个处理元件被配置为,在一系列连续时钟循环期间,从可配置网络电路迭代地接收与数据符号有关的先验软判决值(先验LLR),其中该先验软判决值与表示上部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联,以使用该先验软判决值执行与该窗口相关联的并行计算,以便生成与该数据符号有关的对应非本征软判决值,。该系列连续时钟循环是执行整个解码处理以便在turbo解码处理的多次迭代中恢复帧的数据符号所需求的多个时钟循环。然后,上部解码器的处理元件向可配置网络电路提供非本征软判决值。上部解码器的至少一个处理元件被配置为相对上部解码器的至少一个其他处理元件执行与不同数量的网格级相关联的窗口的计算。这是因为,例如,与上部卷积编码器的状态之间的可能路径相对应的网格级的数量可以不是处理元件数量的整数倍。

[0093] 下部解码器包括与下部卷积编码器相关联的多个下部处理元件,下部解码器的处理元件中的每一个处理元件被配置为在一系列连续时钟循环期间,从可配置网络电路迭代地接收与数据符号有关的先验软判决值,其中该先验软判决值与表示下部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联,以使用该先验软判决值执行与该窗口相关联的并行计算,以便生成与该数据符号有关的对应非本征软判决值。然后,每个处理元件向可配置网络电路提供非本征软判决值。下部解码器的至少一个处理元件被配置为相对下部解码器的至少一个其他处理元件执行与不同数量的网格级相关联的窗口的计算。

[0094] 可配置网络电路包括网络控制器电路,该网络控制器电路在连续时钟循环期间,迭代地控制可配置网络电路的配置,以通过交织由下部解码器提供的非本征软判决值来为

上部解码器提供先验软判决值，并且通过交织由上部解码器提供的非本征软判决值来为下部解码器提供先验软判决值。由网络控制器控制的由可配置网络电路执行的交织是根据预定调度的，这在一个或多个连续时钟循环的不同循环提供先验软判决值，以避免在相同时钟循环期间向上部或下部解码器的相同处理元件提供不同先验软判决值之间的竞争。

[0095] 因此，根据本技术的示例实施例，上部解码器和下部解码器的每个处理元件执行与其网格的窗口相关联的计算。这意味着每个处理元件正在对与帧的数据符号的一部分相关联且相对应的网格的一部分执行与turbo解码的前向递归和后向递归相关联的计算。作为turbo解码器的任意并行处理的结果，处理元件可以划分上部解码器的网格，而不限制窗口大小到处理元件的映射，尽管可以通过在可用的处理元件之间尽可能多地共享网格级的窗口大小实现更大的解码率。这也意味着帧的大小可以独立于可用于执行turbo解码的处理元件的数量而变化，从而可以动态配置通过划分网格形成的窗口大小。

[0096] 本技术的实施例可以通过以一种方式将可配置网络布置为将先验软判决值从上部解码器提供给下部解码器并且从下部解码器提供给上部解码器，来实现这种任意并行解码，其中在该方式中两者都匹配编码器处执行的交织，而且还避免了在相同时钟循环期间向上部或下部解码器的相同处理元件提供不同先验软判决值之间的竞争，因为处理元件正在为包括多于一个的网格级的窗口执行计算。这是通过预定调度来实现的，该预定调度在至少一个时钟循环中布置来自上部解码器或下部解码器的一个或多个先验软判决值，以例如被延迟一个或多个时钟循环、或因为先验软判决值没有被传送而被跳过。已经接收没有延迟或跳过的先验软判决值的处理元件继续由该处理元件使用在先前迭代中接收的该先验软判决值的先前版本执行的计算的前向递归和后向递归。

[0097] 在一些示例性实施例中，为了将来自上部解码器的非本征软判决值传输为用于下部解码器的先验软判决值，并将来自下部解码器的非本征软判决值传输为用于上部解码器的先验软判决值，可配置网络电路可以包括一个或多个存储器，其用于在经由可配置网络电路进行通信之前存储非本征软判决值，或者在通信之后存储先验软判决值。因此，存储器还可以根据预定调度与可配置网络电路的配置相组合，以保持如果其被跳过以避免竞争则不会被更新(重写)的先验软判决值。因此，在一个或多个时钟循环上存储用于turbo解码处理的迭代的先验软判决值，以执行对网格窗口的计算，这可以重用在特定存储器位置处保持的相同的先验软判决值，为此该处理元件被重用，以避免竞争。至于FPTD[26]的示例，这种妥协可能导致精度降低，但是总体而言，turbo解码器的处理可以产生对数据符号的帧的估计的更快结果。

[0098] 每个处理元件可以正根据用于整数个网格级的前向和后向调度来执行计算，这些计算输出成为用于下部解码器或上部解码器中的另一个解码器的先验软判决值的非本征软判决值。因此，交织器的调度相对于这些计算来确定，并且非本征软判决值被生成并传送到上部和下部解码器中的另一个解码器，该非本征软判决值被调度以避免任何竞争，代价是在通过交织器的传送中引入延迟或删除一些软判决值，并且调度被设计成减少这些竞争。预定调度的设计是为了减少延迟和删除。延迟的影响可能是针对处理元件在没有最先进/最新的非本征/先验软判决值(extrinsic/a priori soft decision values)的情况下继续前向递归和后向递归的计算，因为作为竞争的结果，最新版本不能被传送。在一些示例中，软判决值可以比要求的更早传送，但是目的是减少针对由于非本征软判决值而错过的

机会的数量。处理元件使用它在先前次迭代中所具有的先验软判决值，或者如果它是第一次迭代，它将其设置为零。

[0099] 在一些示例实施例中，由一个或多个处理元件根据网格的窗口执行的计算可以由其中根据子窗口执行计算和处理、包括一个或多个时钟循环的不同子周期、例如包括窗口的网格状态的两个子窗口形成。在一些示例中，每个窗口的处理被限制到包括网格级的前半向上舍入(first half rounding up)或网格级的后半向上舍入(last half rounding up)的子窗口。在这些子周期和子窗口内，完成前向递归和后向递归，并且然后，如果子周期中的时钟循环数大于子窗口中的网格级的数量，则执行前向递归和后向递归的开始。

[0100] 在一些示例中，作为由处理元件执行以执行turbo解码处理的计算的一部分，根据前向或后向状态度量中的一个生成非本征软判决值，另一个从存储器加载。

[0101] 图10提供了任意并行turbo解码器(APTD)的示意框图。如图10中所示，并且对应于图7和9所示的参考编号，APTD采用上部解码器1001和下部解码器1002，每个解码器包括P个处理元件的链，其中数字P可以任意选择，诸如P＝64。上部和下部解码器每个包括上部处理元件(PE)1006和下部处理元件1008，它们分别执行与上部卷积编码器和下部卷积解码器相对应的前向递归和后向递归的计算。上部解码器1001和下部解码器1002的每个处理集中的最后一个元件是终止元件1010、1012。

[0102] 如将在以下段落中解释的，上部处理元件1006和下部处理元件1008每个执行计算以实施APTD。然而，为了容纳其中上部处理元件1006和下部处理元件1008每个对多个帧长度执行计算的布置，APTD包括可配置交织器1020，用于将上部解码器1001中的处理元件1006连接到下部解码器1002中的处理元件1008，以及解交织器，用于将下部解码器1002中的每一个处理元件1008连接到上部解码器1001中的每一个处理元件1006。交织器和解交织器各自由包括$S=S(P)=2\lfloor P/2\rfloor+S(\lceil P/2\rceil)+S(\lfloor P/2\rfloor)$个交叉开关(crossbar switch)的Beneš网络形成，其中S(1)＝0且S(2)＝1[25]。例如，当P＝64时S＝352。

[0103] 可配置交织器由两个Beneš网络1022、1024形成，这两个Beneš网络1022、1024由交织器ROM控制器1026结合两个只读存储器1028、1030来控制。交织器ROM控制器1026由计数器1032和控制线驱动，以控制对由上部处理元件集1001、下部处理元件集1002产生的软判决值的Beneš网络切换，使得这些软判决值在能够根据本解码技术优化帧的解码的时间时对每个处理元件可用。最后，如图10中所示，APTD包括CRC单元，以及上部和下部终止元件。

[0104] APTD可用于一次解码一帧比特，支持所有L＝188帧长度$\{K_1,K_2,K_3,\ldots,K_{188}\}=\{40,48,\ldots,5661,4$和LTE turbo码[1]的对应交织器设计。为了发起帧的解码，使用$\lceil\log_2(L)\rceil=8$比特将其长度K_l的索引$l\in[1,L]$输入到APTD，如图10中的1034所示。同时，上部解码器被提供有奇偶校验LLR$[b_{2,k}^{u,a}]_{k=1}^{K_l}$和系统LLR$[b_{3,k}^{u,a}]_{k=1}^{K_l}$。同时，下部解码器被提供有奇偶校验LLR$[b_{2,k}^{l,a}]_{k=1}^{K_l}$。同样，上部终止元件被提供有六个终止LLR$[b_{2,k}^{u,a}]_{k=K_{l+1}}^{K_l+3}$和$[b_{3,k}^{u,a}]_{k=K_{l+1}}^{K_l+3}$，而下部终止元件被提供有再六个终止LLR$[b_{2,k}^{l,a}]_{k=K_{l+1}}^{K_l+3}$和$[b_{3,k}^{l,a}]_{k=K_{l+1}}^{K_l+3}$。

[0105] 终止元件在解码处理开始时仅操作一次。相比之下，上部解码器和下部解码器的处理元件贯穿解码处理连续地操作。更具体地，上部解码器连续地更新K_1个后验LLR$[b_{1,k}^{u,p}]_{k=1}^{K_l}$和$K_1$个非本征LLR$[b_{1,k}^{u,e}]_{k=1}^{K_l}$的值。对非本征LLR的这些更新由第一Beneš网络连续地交织，并作为K_1个先验LLR$[b_{1,k}^{l,a}]_{k=1}^{K_l}$被提供给下部解码器，如图10中所示。更具体地，$b_{1,\Pi(k)}^{l,a}=b_{1,k}^{u,e}$，其中交织动作由向量Π描述，其中$K_1$个元件Π(k)∈[1,$K_1$]中的每一个是唯一的，如以上对$K_1$=40比特LTE交织器的解释中所例示的，该交织器可以由向量Π=[1,38,15,12,29,26,3,40,17,14,31,28,5,2,19,16,33,30,7,4,21,18,35,32,9,6,23,20,37,34,11,8,25,22,39,36,13,10,27,24]来描述，其中$b_{1,24}^{l,a}=b_{1,40}^{u,e}$。同时，下部解码器连续地更新$K_1$个非本征LLR$[b_{1,k}^{l,e}]_{k=1}^{K_l}$的值，这些值由第二Beneš网络解交织，并作为$K_1$个先验LLR$[b_{1,k}^{u,a}]_{k=1}^{K_l}$被提供给上部解码器。更具体地，$b_{1,\Pi^{-1}(k)}^{u,a}=b_{1,k}^{l,e}$，其中解交织动作由向量$\Pi^{-1}$描述，其中$K_1$个元件$\Pi^{-1}$(k)∈[1,$K_1$]中的每一个是唯一的，并且服从与交织器的关系$\Pi^{-1}$(Π(k))=k。同时，对后验LLR$[b_{1,k}^{u,p}]_{k=1}^{K_l}$的更新被连续地提供给CRC单元。一旦使用硬判决$b_{1,k}^{u,p}>0$获得的解码的比特满足LTE CRC，这就停止解码处理。然后，解码的比特和后验LLR由APTD输出，并且下一帧的解码可以开始。

[0106] 在解码处理中，奇偶校验、系统、先验、非本征和后验LLR被分组到连续窗口的链中，其中每个窗口由对应的上部或下部解码器中的连续处理元件处理。更具体地，对于长度为K_1≤2P的短帧，每个解码器中的前(P-K_1/2)个处理元件被去激活，同时剩余的K_1/2个连续处理元件处理连续窗口，每个窗口包括两个每种类型的LLR。因此，由具有索引p∈[1,P]的处理元件处理的每种类型的LLR的数量由以下给出

[0107] $$W_{l,p}=\begin{cases}0 & \text{如果} p \le P-K_l/2\\ 2 & \text{否则}\end{cases}。 \qquad (16)$$

[0108] 等效地，具有索引k∈[1,K_1]的LLR由具有索引$p=\lceil k/2\rceil+P-K_l/2$的对应解码器中的处理元件来处理。相比之下，对于长度为K_1>2P的较长帧，由具有索引p∈[1,P]的处理元件处理的每种类型的LLR的数量由以下给出

[0109] $$W_{l,p}=\begin{cases}\lfloor K_l/P\rfloor & \text{如果} p \le P-\mathrm{mod}(K_l,P)\\ \lceil K_l/P\rceil & \text{否则}\end{cases}。 \qquad (17)$$

[0110] 因此，具有索引k∈[1,K_1]的LLR由具有以下索引的对应解码器中的处理元件来处理

[0111] $$p=\begin{cases}\lceil k/\lfloor K_l/P\rfloor\rceil & \text{如果} k \le k_{\mathrm{edge}},\\ \lceil (k-k_{\mathrm{edge}})/\lceil K_l/P\rceil\rceil+k_{\mathrm{edge}}/\lfloor K_l/P\rfloor & \text{否则}\end{cases} \qquad (18)$$

[0112] 其中$k_{\mathrm{edge}}=[P-\mathrm{mod}(K_l,P)]\cdot\lfloor K_l/P\rfloor$是属于具有长度$\lfloor K_l/P\rfloor$的窗口的最后LLR的索引。这样，每种类型的所有$[P-\mathrm{mod}(K_l,P)]\cdot\lfloor K_l/P\rfloor+\mathrm{mod}(K_l,P)\cdot\lceil K_l/P\rceil=K_l$个LLR都由对应解码器中的P个处理元件来处理。注意，最大窗口长度由$W_{\max}=\max_{l=1}^{L}\max_{p=1}^{P}W_{l,p}=\lceil K_{\max}/P\rceil$

给出，这发生在解码具有$K_{\max}=\max_{l=1}^{L}K_l=6144$比特的最长LTE帧长度的帧时。例如当P＝64时，$W_{\max}$＝96。

[0113] APTD解码处理根据周期性调度完成，其中根据下式，周期C_l取决于当前帧长度K_l，

[0114] $$C_l=\begin{cases}2 & \text{如果}K_l\le 2P\\ K_l/P & \text{如果}\mathrm{mod}(K_l,P)=0,\\ \lceil K_l/P\rceil+D_l & \text{否则}\end{cases} \quad (19)$$

[0115] 其中D_l为非负整数，可以针对每个帧长度K_l单独选择D_l，以便控制纠错能力和APTD的吞吐量之间的权衡。例如，可以针对K_l＝2016选择D_l＝2，其中K_l＝2016是当P＝64时不满足mod(K_l,P)＝0的最长LTE帧长度。在连续的时钟循环中，图10的计数器1026重复计数到C_l，其中使用$X=\lceil\log_2(C_{\max})\rceil$个比特来用信号通知计数器值c，其中。例如，当P＝64且$D_l=2\,\forall l$时，$C_{\max}$＝96并且因此X＝7。

[0116] 贯穿APTD解码处理，当前帧长度K_l的索引l和计数器c的值被提供给每个处理元件，以及图10的交织器ROM控制器1026。交织器ROM控制器将l和c转换成地址，该地址可用于读取Beneš网络的交叉切换模式以在当前时钟循环中使用。如图10中所示，交织器ROM包括可以使用$Y=\lceil\log_2(Z)\rceil$个比特来寻址的$Z=\sum_{l=1}^{188}C_l$个存储器地址。例如，当P＝64和$D_l=2\,\forall l$时，Z＝5,952，并且因此Y＝13。每个交织器ROM中的每个存储器地址存储S个比特，该S个比特为对应Beneš网络中的S个交叉开关中每一个选择切换位置。

[0117] 图10的每个终止元件被提供有六个输入LLR$[b_{2,k}^{a}]_{k=K_l+1}^{K_l+3}$和$[b_{3,k}^{a}]_{k=K_l+1}^{K_l+3}$。这里，上标“u”和“1”已从符号中移除，因为本部分的讨论同样适用于上部和下部终止元件。如图10中所示，每个终止元件输出后向状态度量向量$\beta_{K_l}=[\beta_{K_l}(s_{K_l})]_{s_{K_l}=1}^{S}$，其中在LTE turbo码中S＝8。对于每个后向状态度量$\beta_{K_l}(s_{K_l})$采用固定点二进制表示，其中采用$\beta_{K_l}(s_{K_l})=0$的值直到终止元件的计算完成。更具体地，可以根据(20)获得β_{K_l}的更新值。

[0118] $$\beta_{k-1}(s_{k-1})=\max_{\{s_k|c(s_{k-1},s_k)=1\}}\left[b_2(s_{k-1},s_k)\cdot b_{2,k}^{a}+b_3(s_{k-1},s_k)\cdot b_{3,k}^{a}+\beta_k(s_k)-\beta_k(1)\right] \quad (20)$$

[0119] 该等式是使用使用$\beta_{K_l+3}=[0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty]$初始化的后向递归计算的，其中可以使用由固定点数表示支持的最负值来表示-∞。更具体地，后向递归相继计算β_{K_l+2}、β_{K_l+1}、以及然后最后的输出β_{K_l}。注意，如上所述，符号$b_1(s_{k-1},s_k)$、$b_2(s_{k-1},s_k)$、$b_3(s_{k-1},s_k)$、和$c(s_{k-1},s_k)$由图6所示的示例网格示出。

[0120] 图11提供了示意框图，其示出了形成上部解码器1001和下部解码器1002的图10所示的处理元件的一个示例实施方式的部分。如上所述，处理元件可以被赋予索引p∈[1,P]，使得每个解码器在包括$W_{l,p}$个奇偶校验、系统、先验、非本征和后验LLR的窗口的基础上操作。在本部分中，符号k′∈[1,$W_{l,p}$]用于第p个窗口内的LLR的索引，它可以根据$k=k'+\sum_{p'=1}^{p-1}W_{l,p'}$被转换为帧内的索引k∈[1,$K_l$]。此外，只要本部分的讨论同样适用于上部

和下部解码器的处理元件，就从符号中移除上标“u”和“l”。

[0121] 如图11中所示，处理元件包括执行计算以实施APTD所需求的组件。如图11中所示，处理元件在上侧接收导体(conductor)1101、1102上的当前LLR值，这些值被存储在RAM 1104、1105中。相对应地，在处理元件的下侧，先验LLR值在导体1104上从Beneš网络传递，并被放置在存储RAM 1110中。后向状态度量值在导体1112上被接收并被馈送到复用器1114，该复用器1114利用复用器1116、加法器1118和后向子处理器元件1120执行后向递归的处理计算，以经由复用器1124从导体1122输出对应的后向状态度量值。相对应地，前向状态度量值在导体1130上被接收并被馈送到复用器1132，该复用器1132与复用器1134、加法器1136和第一前向子处理元件1138相组合来执行前向状态度量值的计算，该前向状态度量值经由复用器1142被馈送到输出导体1140。第二前向子处理元件1150接收来自第一前向子处理元件1138的部分计算，并计算经由存储RAM 1108被馈送到输出导体1106上的非本征LLR输出值。与ROM 1172、1162组合的输入控制器1160和输出控制器1170用于控制RAM 1108、1110，以导体1164上从Benes网络接收先验LLR值以及从导体1106输出非本征LLR值两者。类似地，用于前向递归处理和后向递归处理的RAM控制器1174、1176与随机存取存储器1178一起操作，以将后向状态度量值馈送到第一前向子处理元件1138。

[0122] 如将从图11中所示的电路图中理解，与上部解码器不同，下部解码器不受益于系统LLR，这相当于使得$b^{a}_{3,k}=0$。同样，下部解码器不生成后验LLR $b^{p}_{1,k}$。这些差异允许从下部解码器的处理元件中省略用于存储$b^{a}_{3,k}$的RAM，以及与$b^{a}_{3,k}$和$b^{p}_{1,k}$相关联的所有电路。

[0123] 在解码处理开始时，每个解码器中具有索引p的处理元件被提供有$W_{l,p}$个奇偶校验LLR$[b^{a}_{2,k'}]^{W_{l,p}}_{k'=1}$。此外，在解码处理开始时，上部解码器中的第p个处理元件被提供有$W_{l,p}$个系统LLR$[b^{u,a}_{3,k'}]^{W_{l,p}}_{k'=1}$。贯穿解码处理，每个解码器的第p个处理元件被连续地提供有对窗口中先验LLR$[b^{a}_{1,k'}]^{W_{l,p}}_{k'=1}$的更新。作为响应，如下所述，该处理元件连续地更新窗口中的非本征LLR$[b^{e}_{1,k'}]^{W_{l,p}}_{k'=1}$。此外，在上部解码器中的第p个处理元件的情况下，窗口中的后验LLR$[b^{u,p}_{1,k'}]^{W_{l,p}}_{k'=1}$也随着解码处理的进行而被更新。

[0124] 此外，如图10中所示，贯穿解码处理，每个解码器中的第(p-1)个处理元件周期性地向第p个处理元件提供对用于k′＝0的情况的前向状态度量向量$\alpha_{k'}=[\alpha_{k'}(s_{k'})]^{S}_{s_{k'}=1}$的更新。然而，如果第(p-1)个处理元件由于$K_l\leq 2P$而被去激活或者如果p＝1并且因此第(p-1)个处理元件不存在，则这是例外。在这些情况下，第p个处理元件使用图11中所示的复用器1132来采用$\alpha_0=[0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty]$。这里，可以使用由固定点数表示支持的最负值来表示-∞。同样，如图10中所示，每个解码器中的第(p+1)个处理元件周期性地向第p个处理元件提供用于k′＝$W_{l,p}$的情况的后向状态度量向量$\beta_{k'}=[\beta_{k'}(s_{k'})]^{S}_{s_{k'}=1}$的更新。然而，如图10中所示且如上所述，如果p＝P则这是例外，在这种情况下，后向状态度量向量由对应的终止元件提供。如下所述，第p个处理元件周期性地更新分别提供给第(p+1)和(p-1)个处理元件的、对于k′＝$W_{l,p}$的前向状态度量向量$\alpha_{k'}$、以及对于k′＝0的后向状态度量向量$\beta_{k'}$。

[0125] 在解码处理的每个时钟循环期间，如图11中所示，每个处理元件接受上述输入，从RAM读取，在后向、第一前向和第二前向子处理元件内执行处理，写入到RAM以及生成上述输出。每个上部处理元件使用五个RAM，以用于存储F比特奇偶校验LLR $[b^{a}_{2,k'}]^{W_{l,p}}_{k'=1}$、F比特系统LLR $[b^{a}_{3,k'}]^{W_{l,p}}_{k'=1}$、G比特后向状态度量向量 $[\beta_{k'}]^{W_{l,p}}_{k'=1}$、I比特先验LLR $[b^{a}_{1,k'}]^{W_{l,p}}_{k'=1}$ 和I比特非本征LLR $[b^{e}_{1,k'}]^{W_{l,p}}_{k'=1}$。相比之下，每个下部处理元件仅使用四个RAM，因为下部解码器不受益于系统LLR。如上所述，这些RAM中的每一个都包括与APTD支持的最长窗口长度相对应的W_{max}个存储器地址。如图11中所示，RAM可以使用 $V=\lceil \log(W_{max}) \rceil$ 个比特来寻址。例如，当P＝64时，V＝7。

[0126] 存储先验LLR $[b^{a}_{1,k'}]^{W_{l,p}}_{k'=1}$、奇偶校验LLR $[b^{a}_{2,k'}]^{W_{l,p}}_{k'=1}$ 和系统LLR $[b^{a}_{3,k'}]^{W_{l,p}}_{k'=1}$ 的RAM有两个读取端口，在图11中被标记D^{r}_{1}和D^{r}_{2}。这些RAM中的每一个的第二端口由后向RAM控制器$A^{b}\in[1, W_{l,p}]$的输出来寻址A^{r}_{2}，以便向后向子处理元件提供LLR $b^{a}_{1,A^{b}}$、 $b^{a}_{2,A^{b}}$和$b^{a}_{3,A^{b}}$，如图11中所示。注意，加法器还用于向后向子处理元件提供由$(b^{a}_{2,A^{b}}+b^{a}_{3,A^{b}})$给出的LLR。同时，每个随机存取存储器的第一端口由前向RAM控制器的输出来寻址A^{r}_{1}，以便向第一前向子处理元件提供LLR $b^{a}_{1,A^{f}}$、 $b^{a}_{2,A^{f}}$、 $b^{a}_{3,A^{f}}$ 和$(b^{a}_{2,A^{f}}+b^{a}_{3,A^{f}})$。还由对应RAM D^{r}的读取端口向该子处理元件提供后向状态度量向量$\beta_{A^{f}}$，该对应RAM D^{r}的读取端口也由前向RAM控制器A^{f}的输出来寻址A^{r}。同时，该RAM的写入端口D^{w}由后向RAM控制器A^{b}的输出来寻址A^{w}，以便存储在先前时钟循环期间生成的后向状态度量向量$\beta_{A^{b}}$。更具体地，复用器1114被用于在$A^{b}=W_{l,p}$时选择由相邻处理元件或终止元件提供的后向状态度量向量$\beta_{W_{l,p}}$，或者另外选择由后向子处理元件提供的后向状态度量向量$\beta_{W_{l,p}}$。

[0127] 后向和前向RAM控制器由l和c驱动，它们分别使用该l和c来产生地址$A^{b}\in[1,W_{l,p}]$和$A^{f}\in[1,W_{l,p}]$。这里，生成地址A^{b}和A^{f}，使得后向子处理元件和第一前向子处理元件根据图12中所示的调度来操作。这里，后向子处理元件执行后向递归，其中递减的LLR索引k′在连续的时钟循环中被处理。同时，第一前向子处理元件执行前向递归，其中递增的LLR索引k′在连续的时钟循环中被处理。在C_l个时钟循环的每个周期的前$\lfloor C_l/2 \rfloor$个时钟循环期间，上部解码器的这些子处理元件处理前$\lfloor W_{l,p}/2 \rfloor$个LLR索引，而下部解码器的对应子处理元件处理后$\lfloor W_{l,p}/2 \rfloor$个LLR索引。注意，在为提高APTD纠错能力而选择$D_l>0$的情况下，$\lfloor C_l/2 \rfloor$可以大于$\lfloor W_{l,p}/2 \rfloor$。在这些情况中，如图12中所示，前向递归和后向递归的开始在它们第一次完成后重复。在每个周期的后$\lfloor C_l/2 \rfloor$个时钟循环期间，下部解码器的后向子处理元件和第一前向子处理元件处理前$\lceil W_{l,p}/2 \rceil$个LLR索引，而上部解码器的对应子处理元件处理后$\lceil W_{l,p}/2 \rceil$个LLR索引。再次，当$D_l>0$时，递归的开始在它们第一次完成后重复。

[0128] 图12提供了示出由第一和第二前向子处理元件以及与图11中所示的那些相对应

的上部解码器的处理元件的后向子处理元件执行一系列处理序列的图形表示。图12示出了用于操作第一和第二前向子处理元件、以及上部解码器的处理元件内和下部解码器的处理元件内的后向子处理元件的调度。注意，如下所述，第二前向子处理元件的操作相对于第一前向子处理元件的操作延迟一个时钟循环，因为它们被图11中的流水线寄存器分隔开。

[0129] 注意，用于子处理元件的操作的调度可以用矩阵来描述。在上部解码器和下部解码器中的每一个采用P＝9个处理元件来对K_1＝40比特的最短LTE帧长度执行处理的情况下，我们根据(17)获得窗口长度。假设选择D_1＝2以便在纠错能力和吞吐量之间进行权衡，我们根据(19)获得$C_l=\lceil 40/9\rceil+1=7$个时钟循环的周期。在这种情况下，用于上部解码器的第一前向子处理元件的调度可以由如下矩阵描述

[0130] $$\mathbf{K}_l^{\mathrm{ufl}}=\begin{bmatrix}1&5&9&13&17&21&26&31&36\\2&6&10&14&18&22&27&32&37\\1&5&9&13&17&21&26&31&36\\3&7&11&15&19&23&28&33&38\\4&8&12&16&20&24&29&34&39\\3&7&11&15&19&25&30&35&40\\4&8&12&16&20&23&28&33&38\end{bmatrix} \quad (21)$$

[0131] 该矩阵包括针对P＝9个处理元件中的每一个处理元件的一列和针对每个调度周期中的C_1＝6个时钟循环中的每一个时钟循环的一行。这里，如图12中所示，第p列和第c行中的元件标识了在每个调度周期内的第c个时钟循环中由第p个处理元件处理的LLR的索引$k\in[1,K_1]$。同样，用于上部解码器的后向子处理元件、下部解码器的第一前向子处理元件和下部解码器的后向子处理元件的调度可以分别由以下三个矩阵描述。

[0132] $$\mathbf{K}_l^{\mathrm{ub}}=\begin{bmatrix}2&6&10&14&18&22&27&32&37\\1&5&9&13&17&21&26&31&36\\2&6&10&14&18&22&27&32&37\\4&8&12&16&20&25&30&35&40\\3&7&11&15&19&24&29&34&39\\4&8&12&16&20&23&28&33&38\\3&7&11&15&19&25&30&35&40\end{bmatrix} \quad (22)$$

[0133] $$\mathbf{K}_l^{\mathrm{lfl}}=\begin{bmatrix}3&7&11&15&19&24&29&34&39\\4&8&12&16&20&25&30&35&40\\3&7&11&15&19&24&29&34&39\\1&5&9&13&17&21&26&31&36\\2&6&10&14&18&22&27&32&37\\1&5&9&13&17&23&28&33&38\\2&6&10&14&18&21&26&31&36\end{bmatrix} \quad (23)$$

[0134] $$\mathbf{K}_l^{\mathrm{lb}}=\begin{bmatrix}4&8&12&16&20&25&30&35&40\\3&7&11&15&19&24&29&34&39\\4&8&12&16&20&25&30&35&40\\2&6&10&14&18&23&28&33&38\\1&5&9&13&17&22&27&32&37\\2&6&10&14&18&21&26&31&36\\1&5&9&13&17&23&28&33&38\end{bmatrix} \quad (24)$$

[0135] 如图11中所示，除了LLR $b^{\mathrm{a}}_{1,\mathrm{A^b}}$、 $b^{\mathrm{a}}_{2,\mathrm{A^b}}$、 $b^{\mathrm{a}}_{3,\mathrm{A^b}}$和$(b^{\mathrm{a}}_{2,\mathrm{A^b}}+b^{\mathrm{a}}_{3,\mathrm{A^b}})$和之外，后向子处理元件被提供有与由上述复用器1114提供的相同的后向状态度量向量$\beta_{\mathrm{A^b}}$。后向子处理元件使用图13(a)的示意图在(20)的基础上操作，以便生成后向状态度量向量$\beta_{\mathrm{A^b-1}}$。

[0136]

$$\beta_{k-1}(s_{k-1})=\max_{\{s_k|c(s_{k-1},s_k)=1\}}\left[b_1(s_{k-1},s_k)\cdot b^{\mathrm{a}}_{1,k}+b_2(s_{k-1},s_k)\cdot b^{\mathrm{a}}_{2,k}+b_3(s_{k-1},s_k)\cdot b^{\mathrm{a}}_{3,k}+\beta_k(s_k)-\beta_k(1)\right] \quad (25)$$

[0137] 注意，与上部解码器不同，下部解码器不受益于系统LLR，这相当于使得$b^{\mathrm{a}}_{3,k}=0$。这允许在下部解码器的情况下从(25)中省略对应的术语。

[0138] 图13(a)示出了在后向子处理元件中执行(20)的后向状态度量向量$\beta_{k-1}=[\beta_{k-1}(s_{k-1})]^{M}_{s_{k-1}=1}$的计算的处理电路的示意表示。注意，与上部解码器不同，下部解码器不受益于系统LLR。这相当于使得$b^{\mathrm{a}}_{3,k}=0$，这允许从下部解码器的处理元件中省略图13(a)中所示的加法器中的一个。如图13(a)中所示，计算使用通常形成在集成电路上的元件来实施，诸如加法器1301、减法电路1302、限幅电路1304和最大输出电路1306。如将从根据对这些电路的传统操作的理解的电路布局中理解，各个电路实施由图11的处理元件中所示的后向子处理元件执行的计算。

[0139] 如图11中所示，在后向子处理元件的计算后，后向状态度量向量$\beta_{\mathrm{A^b-1}}$被存储在内部寄存器中，以便促进其在下一时钟循环中用作$\beta_{\mathrm{A^b}}$。此外，在$\mathrm{A^b}=1$或者$\mathrm{A^b}=\frac{W_{l,p}}{2}+1$的情况下，后向状态度量向量$\beta_{\mathrm{A^b-1}}$被存储在输出寄存器中，以便分别提供$\beta_0$和$\beta_{W_{l,p}/2}$，以初始化相邻处理元件中的后向递归、或由相同处理元件处理的窗口的另一半中的后向递归。

[0140] 类似地，如上所述，图11的第一前向子处理元件被提供有LLR $b^{\mathrm{a}}_{1,\mathrm{A^f}}$、$b^{\mathrm{a}}_{2,\mathrm{A^f}}$、 $b^{\mathrm{a}}_{3,\mathrm{A^f}}$和$(b^{\mathrm{a}}_{2,\mathrm{A^f}}+b^{\mathrm{a}}_{3,\mathrm{A^f}})$，以及后向状态度量向量$\beta_{\mathrm{A^f}}$。此外，复用器1132用于向第一前向子处理元件提供前向状态度量向量$\alpha_{\mathrm{A^f-1}}$，类比于向后向子处理元件提供$\beta_{\mathrm{A^b}}$，但是具有如上所述提供向量$\alpha_0=[0,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty,-\infty]$的附加选项。第一前向子处理元件使用图13(b)的示意图执行(26)的计算，以便生成前向状态度量向量$\alpha_{\mathrm{A^f}}$。

[0141]

$$\alpha_k(s_k) = \max_{\{s_{k-1}|c(s_{k-1},s_k)=1\}}\left[b_1(s_{k-1},s_k)\cdot b_{1,k}^{a} + b_2(s_{k-1},s_k)\cdot b_{2,k}^{a} + b_3(s_{k-1},s_k)\cdot b_{3,k}^{a} + \alpha_{k-1}(s_{k-1}) - \alpha_{k-1}(1)\right] \quad (26)$$

[0142] 注意，与上部解码器不同，下部解码器不受益于系统LLR，这相当于使得$b_{3,k}^{a}=0$。这允许在下部解码器的情况下从(26)中省略对应的术语。

[0143] 图13(b)示出了在第一前向子处理元件中执行(26)的前向状态度量向量$\alpha_k=[\alpha_k(s_k)]_{s_k=1}^{M}$的计算的处理电路的示意表示。注意，与上部解码器不同，下部解码器不受益于系统LLR。这相当于使得$b_{3,k}^{a}=0$，这允许从下部解码器的处理元件中省略图13(b)中所示的加法器中的一个。如图13(b)中所示，计算使用通常形成在集成电路上的元件来实施，诸如加法器1301、减法电路1302、限幅电路1304和最大输出电路1306。如将从根据对这些电路的传统操作的理解的电路布局中理解，各个电路实施由图11的处理元件中所示的第一前向子处理元件执行的计算。

[0144] 如图11中所示，在第一前向子处理元件的计算之后，前向状态度量向量α_{A^f}被存储在内部寄存器中，以便促进其在下一时钟循环中用作$\alpha_{A^f_{-1}}$。此外，在$A^f=W_{1,p}$或$A^f=W_{1,p}/2$的情况下，前向状态度量向量α_{A^f}被存储在输出寄存器中，以便分别提供$\alpha_{W_{l,p}}$和$\alpha_{W_{l,p}/2}$以初始化相邻处理元件中的前向递归或者由相同处理元件处理的窗口的另一半中的前向递归。

[0145] 同时，第一前向子处理元件使用图14a的示意图执行(27)的计算，以便生成比特度量向量$\boldsymbol{\epsilon}_{A^f}^{c}=\left[\epsilon_{A^f}^{c}(b'_{1,A^f},b'_{2,A^f})\right]_{b'_{2,A^f}\in\{0,1\}}^{b'_{1,A^f}\in\{0,1\}}$。

[0146]

$$\epsilon_k^c(b'_{1,k},b'_{2,k}) = \left[\max_{\left\{(s_{k-1},s_k)\middle|\begin{matrix}b_1(s_{k-1},s_k)=b'_{1,k}\\ b_2(s_{k-1},s_k)=b'_{2,k}\end{matrix}\right\}}\left[\alpha_{k-1}(s_{k-1})+\beta_k(s_k)\right]\right] + b'_{2,k}\cdot b_{2,k}^{a} \quad (27)$$

[0147] 这里，(27)中的上标“c”表示时钟循环索引，其被包括以强调(28)和(29)中流水线的动作，如将在下面讨论的。

[0148] 图14(a)提供了示意电路图，该示意电路图提供了用于计算第一前向子处理元件中(27)的比特度量向量$\boldsymbol{\epsilon}_{A^f}^{c}=\left[\epsilon_{A^f}^{c}(b'_{1,A^f},b'_{2,A^f})\right]_{b'_{2,A^f}\in\{0,1\}}^{b'_{1,A^f}\in\{0,1\}}$的电路。再次，图14a中所示的电路执行诸如加法器1401、最大值形成器1402和复用器1404的集成电路元件的传统功能。

[0149] 如图11中所示，流水线寄存器用于向第二前向子处理元件供应比特度量向量$\varepsilon^{c}_{\mathrm{A^f}}$，以及LLR$b^{\mathrm{a}}_{1,\mathrm{A^f}}$和$b^{\mathrm{a}}_{3,\mathrm{A^f}}$，尽管相对于第一前向子处理元件延迟了一个时钟循环。第二前向子处理元件在图14(b)中所示的示意图的基础上工作。在上部解码器中，如上所述，这用于生成由处理元件输出并提供给CRC单元的、(29)的后验LLR$b^{\mathrm{p}}_{1,\mathrm{A^f}}$。在上部和下部解码器两者中，第二前向子处理元件计算提供给对应RAM的写入端口$\mathrm{D^w}$并被放置在由流水线寄存器驱动的地址$\mathrm{A^w}$中的、(28)的非本征LLR，该流水线寄存器供应$\mathrm{A^f}$的延迟副本，如图11中所示。

[0150] $$b^{\mathrm{e}}_{1,k}=0.75\left[\max_{b'_{2,k}\in\{0,1\}}[\varepsilon^{c-1}_{k}(1,b'_{2,k})]\right]-0.75\left[\max_{b'_{2,k}\in\{0,1\}}[\varepsilon^{c-1}_{k}(0,b'_{2,k})]\right]+b^{\mathrm{a}}_{3,k} \qquad (28)$$

[0151] $$b^{\mathrm{p}}_{1,k}=0.75\left[\max_{b'_{2,k}\in\{0,1\}}[\varepsilon^{c-1}_{k}(1,b'_{2,k})]\right]-0.75\left[\max_{b'_{2,k}\in\{0,1\}}[\varepsilon^{c-1}_{k}(0,b'_{2,k})]\right]+b^{\mathrm{a}}_{1,k}+b^{\mathrm{a}}_{3,k} \qquad (29)$$

[0152] 注意，与上部解码器不同，下部解码器不受益于系统LLR，这相当于使得$b^{\mathrm{a}}_{3,k}=0$。这允许在下部解码器的情况下从(28)中省略对应的术语。同样，下部解码器不生成后验LLR，允许(29)被完全省略。这里，(28)和(29)中的上标“c-1”表示先前时钟循环的索引，以便强调(27b)的比特度量向量已经被流水线化。

[0153] 图14b提供了示意电路图，该示意电路图提供了用于计算第二前向子处理元件中(28)和(29)的非本征LLR$b^{\mathrm{e}}_{1,k}$和后验LLR$b^{\mathrm{p}}_{1,k}$的电路。注意，与上部解码器不同，下部解码器不受益于系统LLR，这相当于使得$b^{\mathrm{a}}_{3,k}=0$。同样，下部解码器不生成后验LLR$b^{\mathrm{p}}_{1,k}$。这些差异允许从下部解码器的处理元件中省略图14b中所示的加法器中的三个加法器。再次，图14b中所示的电路执行诸如减法1410、乘法1408和限幅电路1406的集成电路元件的传统功能。

[0154] 注意，对于第二前向子处理元件的调度可以由以上在(21)和(23)中对于第一前向子处理元件例示的相同矩阵来描述，但是由于流水线延迟而向下旋转(rotate)一行。在该示例中，对于上部和下部解码器中的第二前向子处理元件的调度可以分别由以下两个矩阵来描述。

[0155] $$\mathbf{K}^{\mathrm{uf2}}_{l}=\begin{bmatrix}4&8&12&16&20&23&28&33&38\\1&5&9&13&17&21&26&31&36\\2&6&10&14&18&22&27&32&37\\1&5&9&13&17&21&26&31&36\\3&7&11&15&19&23&28&33&38\\4&8&12&16&20&24&29&34&39\\3&7&11&15&19&25&30&35&40\end{bmatrix} \qquad (30)$$

[0156] $$\mathbf{K}_l^{\mathrm{lf2}}=\begin{bmatrix}2&6&10&14&18&21&26&31&36\\3&7&11&15&19&24&29&34&39\\4&8&12&16&20&25&30&35&40\\3&7&11&15&19&24&29&34&39\\1&5&9&13&17&21&26&31&36\\2&6&10&14&18&22&27&32&37\\1&5&9&13&17&23&28&33&38\end{bmatrix} \quad (31)$$

[0157] 在通过交织器和解交织器的LLR交换与前向递归一起被调度的方案中，存储非本征LLR$[b_{1,k'}^{e}]_{k'=1}^{W_{l,p}}$的RAM将表现得像附加的流水线寄存器。更具体地，由第二前向子处理元件在特定时钟循环中写入该RAM的LLR将在下一个时钟循环通过交织器或解交织器来被读取和交换。在这种情况下，对于交织和解交织调度的调度可以由以上在(30)和(31)中对于第一前向子处理元件例示的矩阵来描述，但是由于流水线延迟而向下旋转两行。在这个示例中，由上部解码器向交织器和由下部解码器向解交织器提供非本征LLR的调度可以分别由以下两个矩阵来描述。

[0158] $$\mathbf{K}_l^{\mathrm{u}\pi}=\begin{bmatrix}3&7&11&15&19&25&30&35&40\\4&8&12&16&20&23&28&33&38\\1&5&9&13&17&21&26&31&36\\2&6&10&14&18&22&27&32&37\\1&5&9&13&17&21&26&31&36\\3&7&11&15&19&23&28&33&38\\4&8&12&16&20&24&29&34&39\end{bmatrix} \quad (32)$$

[0159] $$\mathbf{K}_l^{\mathrm{l}\pi}=\begin{bmatrix}1&5&9&13&17&23&28&33&38\\2&6&10&14&18&21&26&31&36\\3&7&11&15&19&24&29&34&39\\4&8&12&16&20&25&30&35&40\\3&7&11&15&19&24&29&34&39\\1&5&9&13&17&21&26&31&36\\2&6&10&14&18&22&27&32&37\end{bmatrix} \quad (33)$$

[0160] 如上所述，$K_1=40$比特LTE交织器和解交织器可以分别由向量$\Pi=[1,38,15,12,29,26,3,40,17,14,31,28,5,2,19,16,33,30,7,4,21,18,35,32,9,6,23,20,37,34,11,8,25,22,39,36,13,10,27,24]$和$\Pi^{-1}=[1,14,7,20,13,26,19,32,25,38,31,4,37,10,3,16,9,22,15,28,21,34,27,40,33,6,39,12,5,18,11,24,17,30,23,36,29,2,35,8]$来描述。因此，在我们的示例中，由交织器向下部解码器和由解交织器向上部解码器提供先验LLR的调度可以分别由以下两个矩阵描述。

[0161] $$\mathbf{\Pi}(\mathbf{K}_l^{u\pi})=\begin{bmatrix}15&3&31&19&7&9&34&39&24\\12&40&28&16&4&35&20&25&10\\1&29&17&5&33&21&6&11&36\\38&26&14&2&30&18&23&8&13\\1&29&17&5&33&21&6&11&36\\15&3&31&19&7&35&20&25&10\\12&40&28&16&4&32&37&22&27\end{bmatrix} \quad (34)$$

[0162] $$\mathbf{\Pi}^{-1}(\mathbf{K}_l^{l\pi})=\begin{bmatrix}1&13&25&37&9&27&12&17&2\\14&26&38&10&22&21&6&11&36\\7&19&31&3&15&40&5&30&35\\20&32&4&16&28&33&18&23&8\\7&19&31&3&15&40&5&30&35\\1&13&25&37&9&21&6&11&36\\14&26&38&10&22&34&39&24&29\end{bmatrix} \quad (35)$$

[0163] 先验LLR被传送到的下部解码器和上部解码器内的特定处理元件可以由矩阵$\mathbf{P}_l^{u\pi}$和$\mathbf{P}_l^{l\pi}$来描述，该矩阵$\mathbf{P}_l^{u\pi}$和$\mathbf{P}_l^{l\pi}$可以分别通过将(18)应用于$\mathbf{\Pi}(\mathbf{K}_l^{u\pi})$和$\mathbf{\Pi}^{-1}(\mathbf{K}_l^{l\pi})$来获得。在我们的示例中，我们获得以下矩阵。

[0164] $$\mathbf{P}_l^{u\pi}=\begin{bmatrix}4&1&\mathbf{8}&5&2&3&\mathbf{8}&9&6\\\mathbf{3}&9&7&4&1&8&5&6&\mathbf{3}\\1&7&5&\mathbf{2}&8&6&\mathbf{2}&3&9\\9&\mathbf{7}&\mathbf{4}&1&\mathbf{7}&5&6&2&\mathbf{4}\\1&7&5&\mathbf{2}&8&6&\mathbf{2}&3&9\\4&1&\mathbf{8}&\mathbf{5}&2&\mathbf{8}&\mathbf{5}&6&3\\3&\mathbf{9}&\mathbf{7}&4&1&8&\mathbf{9}&6&\mathbf{7}\end{bmatrix} \quad (36)$$

[0165] $$\mathbf{P}_l^{l\pi}=\begin{bmatrix}1&4&6&9&\mathbf{3}&7&\mathbf{3}&5&1\\4&7&\mathbf{9}&\mathbf{3}&\mathbf{6}&\mathbf{6}&2&\mathbf{3}&\mathbf{9}\\\mathbf{2}&5&\mathbf{8}&1&4&9&\mathbf{2}&7&\mathbf{8}\\\mathbf{5}&\mathbf{8}&1&4&7&\mathbf{8}&\mathbf{5}&6&2\\\mathbf{2}&5&\mathbf{8}&1&4&9&\mathbf{2}&7&\mathbf{8}\\1&4&\mathbf{6}&\mathbf{9}&\mathbf{3}&\mathbf{6}&2&\mathbf{3}&\mathbf{9}\\4&\mathbf{7}&\mathbf{9}&3&\mathbf{6}&8&\mathbf{9}&\mathbf{6}&\mathbf{7}\end{bmatrix} \quad (37)$$

[0166] 然而，这些矩阵揭示了通过交织器和解交织器的LLR交换与前向递归一起被调度的方案导致了竞争问题。更具体地，在上面提供的示例矩阵$\mathbf{P}_l^{u\pi}$和$\mathbf{P}_l^{l\pi}$中，一些行包含重复的处理元件索引，如以粗体突出显示的。然而，以这种方式，用于实施交织器和解交织器的Beneš网络不能同时向处理元件传送一个以上的LLR。

[0167] 为了解决这种竞争问题，我们独立于前向递归和后向递归来调度交织和解交织。

更具体地，如上所述，图11的前向递归和后向递归是通过使用前向和后向RAM控制器调度大多数RAM的读取和写入操作来实施的。相反，对于存储先验LLR$[b^{a}_{1,k'}]^{W_1,p}_{k'=1}$的RAM的写入端口$D^{W}$和存储非本征LLR$[b^{e}_{1,k'}]^{W_1,p}_{k'=1}$的RAM的读取端口$D^{r}$使用独立的调度。这允许通过交织器和解交织器的LLR的交换被重新调度，以避免每当两个或更多LLR在相同时钟循环中被指定用于相同处理元件时将会引起的Beneš网络竞争。这种重新调度仍然可以在紧接着由第一和第二前向子处理元件生成的两个时钟循环的第三时钟循环中、通过交织器或解交织器传送特定的非本征LLR。这将潜在地允许LLR在第四时钟循环中、被其他解码器的连接的处理元件中的第一前向或后向子处理元件使用。然而，该连接的处理元件的前向递归和后向递归可能需要几个时钟循环来达到这种更新的LLR，在此期间它将不会被使用。这种观察揭示了，延迟通过交织器或解交织器的一些LLR的传送可以不对APTD的操作产生不利影响。受此激发，特定LLR的交织或解交织可以被延迟，以减缓竞争。然而，为了消除竞争，也可以有必要禁用由特定处理元件在调度周期内的特定时钟循环中生成的非本征LLR的交织或解交织。正是这激发了对于特定帧长度K_1采用调度周期内的D_1个附加时钟循环。如上所述，这些附加时钟循环允许每个前向递归的开始在其完成后重复，从而给予第二次机会来交织或解交织可能在第一次机会时被禁用的非本征LLR。通过仔细设计交织器调度，可以确保每个非本征LLR在每个调度周期至少交织或解交织一次，同时最小化延迟它们的传送的不利影响，并最小化每个调度周期采用的附加时钟循环的数量D_1。

[0168] 下面提供了用于设计交织或解交织调度的特定算法。

[0169]

$\mathbf{K}_l^{\pi} \leftarrow \mathbf{K}_l^{\mathrm{fl}}$

$\mathbf{P}_l^{\pi} \leftarrow \mathbf{P}_l^{\mathrm{fl}}$

for $col=1$ 至 P **do**

  for $row=1$ 至 C_l **do**

    if $\mathbf{K}_l^{\pi}(row,col)$复制相同列中更高的元素的值 **then**

      通过将它们的值设置为‘–’来禁用$\mathbf{K}_l^{\pi}(row,col)$和$\mathbf{P}_l^{\pi}(row,col)$

    end if

  end for

end for

for $row=1$ 至 C_l **do**

  for $col=1$ 至 P **do**

    if $\mathbf{P}_l^{\pi}(row,col)$复制相同行中更右的元素的值 **then**

      使用从矩阵底部到顶部换行（wrap）的顺序，在连续的行中搜索第一行，在该第一行中，任何列中的$\mathbf{P}_l^{\pi}(row,col)$值不重复并且列$col$中的元素被设置为‘–’

      if 合适的行被标识 **then**

        通过将它们的值与所标识的行的相同列中的‘–’进行交换来延迟$\mathbf{K}_l^{\pi}(row,col)$和$\mathbf{P}_l^{\pi}(row,col)$

      else

        return ‘未成功’

      end if

        end if

      end for

[0170] **end for**

将$\mathbf{K}_l^{\pi}$向下旋转两行

return ‘成功’

[0171] 为了最大化APTD的吞吐量，以上算法可以相继使用更高的D_1值，直到对于交织器和解交织器两者都成功。在我们的示例中，得到的交织和解交织调度$\mathbf{K}_l^{u\pi}$和$\mathbf{K}_l^{l\pi}$分别由(38)和(39)给出。这里，括号中提供了$\mathbf{P}_l^{u\pi}$和$\mathbf{P}_l^{l\pi}$的对应值，其示出了所有竞争都已被消除。

[0172] $$\mathbf{K}_l^{u\pi}=\begin{bmatrix} - & - & 12(7) & 15(5) & - & 25(3) & 30(8) & 35(9) & 40(6) \\ - & 8(9) & 11(8) & - & - & - & - & - & - \\ 1(1) & 5(7) & 9(5) & - & 17(8) & 21(6) & 26(2) & 31(3) & 36(9) \\ 2(9) & - & - & 14(1) & 18(7) & 22(5) & 27(6) & 32(2) & 37(4) \\ - & 6(7) & 10(4) & 13(2) & - & - & - & - & - \\ 3(4) & 7(1) & - & - & 19(2) & 23(8) & 28(5) & 33(6) & 38(3) \\ 4(3) & - & - & 16(4) & 20(1) & 24(8) & 29(9) & 34(6) & 39(7) \end{bmatrix} \quad (38)$$

[0173] $$\mathbf{K}_l^{l\pi}=\begin{bmatrix} - & 8(8) & 9(6) & 13(9) & - & 23(7) & 28(3) & 33(5) & 38(1) \\ 4(5) & 6(7) & 10(9) & - & 17(3) & - & - & - & - \\ - & 7(5) & - & 15(1) & 19(4) & 24(9) & 29(2) & 34(7) & 39(8) \\ - & - & 12(1) & 16(4) & 20(7) & 25(8) & 30(5) & 35(6) & 40(2) \\ 3(2) & - & 11(8) & - & 18(6) & - & - & - & - \\ 1(1) & 5(4) & - & - & - & 21(6) & 26(2) & 31(3) & 36(9) \\ 2(4) & - & - & 14(3) & - & 22(8) & 27(9) & 32(6) & 37(7) \end{bmatrix} \quad (39)$$

[0174] 如图10中所示，交织器和解交织器调度被存储在ROM中。这些ROM由将1和c转换成地址A^r的输出和输入ROM控制器来控制，该地址A^r可用于从对应ROM的读取端口D^r读取RAM地址。每个ROM包括可以使用$Y=\lceil \log_2(Z) \rceil$个比特来寻址的$Z=\sum_{i=1}^{188} C_l$个存储器地址。每个ROM存储器地址包括$V'=\lceil \log_2(W_{max})+1 \rceil$个比特，其可以存储1到$W'_{max}$范围内的值。这里，$W'_{max}=W_{max}+1$表示用于表示交织器或解交织器调度中的禁用的条目的虚拟(dummy)值，该虚拟值在(38)和(39)中用‘-’来表示。注意，用于从一个处理元件读取非本征LLR的调度必须与用于将先验LLR写入通过交织器或解交织器连接的处理元件的调度相对应。注意，为了最大化APTD的时钟频率，在Beneš网络内包括流水线寄存器可以是有益的。在这种情况下，对于每个流水线级，用于写入先验LLR的调度应该向下旋转一个位置。

[0175] 优点概述

[0176] 如以上所解释的本技术的实施例可以提供具有以下优点的APTD：

[0177] 传统turbo解码器被限制为采用数量为P的处理元件，该数量P是帧长度K_l的整数倍。这确保了所有窗口具有相同的长度$W_l=K_l/P$，并且交织可以在没有竞争的情况下完成。相比之下，APTD支持任意数量P的处理元件，并采用不同长度的窗口。APTD通过独立于非本征LLR的生成来调度该非本征LLR的交织和解交织以避免竞争。更具体地，一些非本征LLR的交织或解交织相对于它们的生成被延迟，或者被完全禁用。

[0178] 当帧长度K_l短于2048比特时，示例2的turbo解码器禁用其P=64个处理元件中的一些。相比之下，当帧长度K_l短于2P时，APTD仅禁用每个解码器中其P个处理元件中的一些。在这种情况下，每个解码器中的$K_l/2$个处理元件处理长度为$W_{l,p}=2$的窗口，而剩余的处理元件被禁用。当K_l大于2P时，窗口中的一些窗口长度为$W_{l,p}=\lfloor K_l/P \rfloor$，而剩余的窗口长度为$W_{l,p}=\lceil K_l/P \rceil$。

[0179] 像示例3的混洗(shuffled)turbo解码器一样，APTD对于上部解码器的每个窗口采

用一个处理元件，以及对于下部解码器的每个窗口采用单独的处理元件，其中贯穿解码处理来同时处理所有窗口。然而，当混洗turbo解码器在每个窗口内执行单个前向递归和单个后向递归时，APTD将每个窗口划分为两个子窗口。APTD在对另一子窗口执行前向递归和后向递归之前，对一个子窗口执行前向递归和后向递归。这根据奇偶布置来执行，使得上部解码器的每个窗口中的第一子窗口与下部解码器的每个窗口中的第二子窗口同时处理，反之亦然。注意，在所有窗口都具有偶数长度$W_{l,p}$的情况下，这种布置相当于使得两倍的窗口，并且使用每个处理元件在相同解码器内的两个相邻窗口的处理之间交替。这与使用每个处理元件在上部解码器中的特定窗口的处理和下部解码器中的对应窗口之间交替的示例1、2和4的turbo解码器形成对比。注意，APTD采用的方案的优点在于消除了对处理元件能够将非本征LLR交织或解交织回自身的要求，允许采用更简单的交织器和解交织器。

[0180] 在APTD的窗口具有最小长度$W_{l,p}=2$的特殊情况下，上述奇偶布置变得相当于FPTD的奇偶布置，并且以相同的方式受益于LTE交织器的奇偶性质。在窗口长度$W_{l,p}$为奇数的情况下，在前$\lfloor C_l/2 \rfloor$个时钟循环期间执行的递归的长度为$\lfloor W_{l,p}/2 \rfloor$，而在剩余$\lfloor C_l/2 \rfloor$个时钟循环中执行的递归的长度为$\lceil W_{l,p}/2 \rceil$，导致为上部解码器和下部解码器执行的递归之间稍微重叠。这与先前提出的turbo解码器的递归形成对比，该先前提出的turbo解码器没有重叠递归。

[0181] 示例1至3的已知turbo解码器在前向递归和后向递归两者的后半部分期间生成非本征LLR。这种方案在递归的前半部分期间不生成非本征LLR，在后半部分的每个步骤产生两个LLR。因此，这种方案在后半部分要求两个交织器和两个解交织器，并且在递归的前半部分该硬件未使用。相比之下，APTD仅在前向递归期间基于已经最近生成的后向状态度量来生成非本征LLR，该后向状态度量是在先前解码迭代期间执行的递归的结束期间、或者在当前递归的开始期间生成的。如图15和16所表征的，这仅允许使用单个交织器和单个解交织器，尽管这实现的代价是要求更多解码迭代以便实现相同比特错误率(BER)。

[0182] 与示例1至3的turbo解码器相反，APTD可以在递归的完成后重复该递归的开始。这提供了第二次机会来生成相关联的非本征LLR，允许禁用这些重新生成的LLR中的一个或其他的交织或解交织，而不完全消除这些LLR的交织或解交织。当窗口长度$W_{l,p}$短时，这也允许生成更近的后向状态度量向量，准备用于在下一次迭代中生成非本征LLR。

[0183] 像示例4的FPTD一样，APTD采用流水线来增加最大时钟频率，但代价是要求更多的解码迭代来实现相同BER。虽然通过FPTD的上部解码器和下部解码器中的一个的流水线有三个级，但APTD将其减少到两个级，从而提高BER。这是通过对每个子处理元件在输入处执行状态度量的归一化和限幅来实现的，而不是像在FPTD那样在输出处执行状态度量的归一化和限幅。

[0184] 说明性结果

[0185] 图15提供了当采用总共2P＝128个处理元件以及最大值8、16、24、32、40、48、56和64个时钟循环来解码具有长度$K_l=512$的帧时、所提出的APTD“提出的”的误码率(BER)性能的曲线图。注意，流水线在这些结果中被禁用，并且当它被启用时，可以预期有轻微的BER下降。这些结果与示例1的turbo解码器的对应版本“基准(仅前向)”进行比较，其仅在前向递归上计算非本征LLR，如在所提出的方案中那样。它采用硬件复杂度与提出的处理元件中的

每一个类似的8个并行处理器，以及最大值246、466、686、909、1126、1346、1566和1786个时钟循环来解码具有相同长度K_1＝512的帧。还提供了示例1的turbo解码器的第二版本“基准(前向和后向)”的结果，其在前向递归和后向递归两者上计算非本征LLR，代价是硬件复杂度比提出的处理元件中的每一个高42％。该方案采用8个并行处理器以及最大值246、466、686、909、1126、1346、1566和1786个时钟循环来解码具有相同长度K_1＝512的帧。

[0186] 图16提供了当采用总共2P＝128个处理元件以及最大值96、192、288、384、480、576、672和768个时钟循环来解码具有长度K_1＝6144的帧时，所提出的APTD“提出的”的误码率(BER)性能的曲线图。注意，流水线在这些结果中被禁用，并且当它被启用时，可以预期有轻微的BER下降。这些结果与示例1的turbo解码器的相应版本“基准(仅前向)”进行比较，其仅在前向递归上计算非本征LLR，如在所提出的方案中那样。它采用硬件复杂度与提出的处理元件中的每一个类似的8个并行处理器，以及最大值1654、3282、4910、6538、8166、9794、11422、13050个时钟循环来解码具有相同长度K_1＝6144的帧。还提供了示例1的turbo解码器的第二版本“基准(前向和后向)”的结果，其在前向递归和后向递归上计算非本征LLR，代价是硬件复杂度比提出的处理元件中的每一个高42％。该方案使用8个并行处理器以及最大值1654、3282、4910、6538、8166、9794、11422、13050个时钟循环来解码具有相同长度K_1＝6144的帧。

[0187] 以下段落提供了本技术的进一步的方面和特征：

[0188] 一种turbo解码器电路，用于执行turbo解码处理以从接收到的信号中恢复数据符号的帧，该数据符号的帧包括该帧的每个数据符号的一个或多个奇偶校验和/或系统软判决值。该帧的数据符号已经用turbo编码器编码，该turbo编码器包括每个能够由网格表示的上部卷积编码器和下部卷积编码器、以及在上部卷积编码器和下部卷积编码器之间交织数据符号的交织器。turbo解码器电路包括时钟、被配置为交织软判决值的可配置网络电路、以及上部解码器和下部解码器。上部解码器包括与上部卷积编码器相关联的多个上部处理元件，上部解码器的处理元件中的每一个处理元件被配置为，在一系列连续时钟循环期间，从可配置网络电路迭代地接收与数据符号有关的先验软判决值，其中该先验软判决值与表示上部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联，以使用该先验软判决值执行与该窗口相关联的并行计算，以便通过执行turbo解码的前向递归和后向递归生成与数据符号有关的对应非本征软判决值，以及被配置为向可配置网络电路提供非本征软判决值，上部解码器的至少一个处理元件被配置为相对上部解码器的至少一个其他处理元件执行与不同数量的网格级相关联的窗口的计算。下部解码器包括与下部卷积编码器相关联的多个下部处理元件，下部解码器的处理元件中的每一个处理元件被配置为，在一系列连续时钟循环期间，从可配置网络电路迭代地接收与数据符号有关的先验软判决值，其中该先验软判决值与表示下部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联，以使用该先验软判决值执行与该窗口相关联的并行计算，以便通过执行turbo解码的前向递归和后向递归生成与数据符号有关的对应非本征软判决值，以及被配置为向可配置网络电路提供非本征软判决值，下部解码器的至少一个处理元件被配置为相对下部解码器的至少一个其他处理元件执行与不同数量的网格级相关联的窗口的计算。可配置网络电路包括网络控制器电路，该网络控制器电路在连续时钟循环期间迭代地控制可配置网络电路的配置，以通过交织由下部解码器提供的非本征软判决值来为上部解

码器提供先验软判决值，并且通过交织由上部解码器提供的非本征软判决值来为下部解码器提供先验软判决值，由网络控制器控制的可配置网络电路执行的交织是根据预定调度的，这在一个或多个连续时钟循环的不同循环提供先验软判决值，以避免在相同时钟循环期间向上部或下部解码器的相同处理元件提供不同先验软判决值之间的竞争。

[0189] 一种turbo解码器电路，用于执行turbo解码处理以从接收到的信号中恢复数据符号的帧，该数据符号的帧包括该帧的每个数据符号的软判决值。该帧的数据符号已经用turbo编码器编码，该turbo编码器包括每个能够由网格表示的上部卷积编码器和下部卷积编码器、以及在上部卷积编码器和下部卷积编码器之间交织数据符号的交织器。turbo解码器电路包括时钟、被配置为交织软判决值的可配置网络电路、以及上部解码器和下部解码器。上部解码器包括与上部卷积编码器相关联的多个上部处理元件，上部解码器的处理元件中的每一个处理元件被配置为在一系列连续时钟循环期间，从可配置网络电路迭代地接收与数据符号有关的先验软判决值，以执行并行计算来生成与数据符号有关的对应非本征软判决值，并向可配置网络电路提供非本征软判决值，其中该先验软判决值与表示上部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联。下部解码器包括与下部卷积编码器相关联的多个下部处理元件，下部解码器的处理元件中的每一个处理元件被配置为在一系列连续时钟循环期间，从可配置网络电路迭代地接收与数据符号有关的先验软判决值，以执行并行计算来生成与数据符号有关的对应非本征软判决值，并向可配置网络电路提供非本征软判决值，其中该先验软判决值与表示下部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联。可配置网络电路根据预定调度进行配置，以在上部和下部解码器之间提供一个或多个连续时钟循环的不同循环的先验软判决值，以避免不同先验软判决值之间的竞争。

[0190] 根据以上段落中所述的实施例，由处理元件根据前向递归和后向递归而执行的计算包括：接收与相邻网格级有关的前向或后向状态度量，将前向或后向状态度量与数据符号的先验、奇偶校验和系统软判决值进行组合，并生成与另一相邻网格级有关的前向或后向状态度量，其中所接收的前向或后向状态度量在与先验、奇偶校验和系统软判决值组合之前被归一化。

[0191] 以下编号的段落提供了示例实施例的进一步示例方面和特征：

[0192] 第1段、一种turbo解码器电路，用于执行turbo解码处理以从接收到的信号中恢复数据符号的帧，该数据符号的帧包括该帧的每个数据符号的一个或多个奇偶校验和/或系统软判决值，该帧的数据符号已经用turbo编码器编码，该turbo编码器包括每个能够由网格表示的上部卷积编码器和下部卷积编码器、以及在上部卷积编码器和下部卷积编码器之间交织该数据符号的交织器，该turbo解码器电路包括：

[0193] 时钟，

[0194] 可配置网络电路，被配置为交织软判决值，

[0195] 上部解码器，包括与上部卷积编码器相关联的多个上部处理元件，上部解码器的处理元件中的每一个处理元件被配置为，在一系列连续时钟循环期间，从可配置网络电路迭代地接收与数据符号有关的先验软判决值，其中该先验软判决值与表示上部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联，以使用先验软判决值执行与该窗口相关联的并行计算以便生成与数据符号有关的对应非本征软判决值，以及被配置为向

可配置网络电路提供非本征软判决值，上部解码器的至少一个处理元件被配置为相对上部解码器的至少一个其他处理元件执行与不同数量的网格级相关联的窗口的计算，以及

[0196] 下部解码器，包括与下部卷积编码器相关联的多个下部处理元件，下部解码器的处理元件中的每一个处理元件被配置为，在该系列连续时钟循环期间，从可配置网络电路迭代地接收与数据符号有关的先验软判决值，其中该先验软判决值与表示下部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联，以使用先验软判决值执行与该窗口相关联的并行计算以便生成与数据符号有关的对应非本征软判决值，以及被配置为向可配置网络电路提供非本征软判决值，下部解码器的至少一个处理元件被配置为相对下部解码器的至少一个其他处理元件执行与不同数量的网格级相关联的窗口的计算，

[0197] 其中可配置网络电路包括网络控制器电路，网络控制器电路在该连续时钟循环期间迭代地控制可配置网络电路的配置，以通过交织由下部解码器提供的非本征软判决值来为上部解码器提供先验软判决值，以及通过交织由上部解码器提供的非本征软判决值来为下部解码器提供先验软判决值，由网络控制器控制的可配置网络电路执行的交织是根据预定调度的，这在一个或多个连续时钟循环的不同循环提供先验软判决值，以避免在相同时钟循环期间向上部或下部解码器的相同处理元件提供不同先验软判决值之间的竞争。

[0198] 第2段、根据段落1所述的turbo解码器电路，其中上部解码器和下部解码器中的每一个的处理元件被配置为从存储器读取先验软判决值，并且在执行计算之后将非本征软判决值写入存储器，并且可配置网络电路被配置为从存储器读取非本征软判决值，并且将先验软判决值写入存储器，并且可配置网络根据预定调度对一个或多个非本征软判决值的读取相对于处理元件对一个或多个非本征软判决值的写入延迟了一个或多个时钟循环。

[0199] 第3段、根据段落1或2所述的turbo解码器电路，其中，上部解码器和下部解码器中的每一个的处理元件被配置为从存储器读取先验软判决值，以及在执行计算之后将非本征软判决值写入存储器，并且可配置网络电路被配置为从存储器读取非本征软判决值，以及将先验软判决值写入存储器，以及由可配置网络根据预定调度对非本征软判决值中的一个或多个非本征软判决值的读取或者由处理元件对一个或多个非本征软判决值的写入中的至少一个被跳过。

[0200] 第4段、根据段落1、2或3中任一段所述的turbo解码器电路，其中，上部解码器或下部解码器中的处理元件的数量不是网格级的数量的整数倍。

[0201] 第5段、根据段落1至4中任一段所述的turbo解码器，其中，由处理元件处理的每个窗口内网格级的最小和最大数量之间的差在上部和下部解码器中的任一个中是一。

[0202] 第6段、根据段落1至5中任一段所述的turbo解码器电路，其中，窗口中的每个窗口包括由彼此相邻的处理元件处理的相同数量的网格级。

[0203] 第7段、根据段落1至6中任一项所述的turbo解码器电路，其中，上部和下部解码器包括相同数量的处理元件，并且上部解码器的每个处理元件如下部解码器的对应处理元件一样对包括对应网格级的窗口执行计算。

[0204] 第8段、根据段落1至7中任一段所述的turbo解码器，其中，处理元件和交织的处理调度根据相同数量的时钟循环是周期性的，每次迭代表示相同调度的周期。

[0205] 第9段、根据段落8所述的turbo解码器，其中，周期由上部或下部解码器中的任一个窗口中的最大网格级加上减少用于根据预定调度跳过以避免竞争的需求而所需的非负

整数来给出。

[0206] 第10段、根据段落1至9中任一段所述的turbo解码器电路，其中，处理元件中的每一个处理元件被配置为根据周期性调度执行并行计算，并且每个周期包括第一子周期和第二子周期，第一子周期包括周期中的一个或多个第一时钟循环，第二子周期包括周期中的剩余循环，其中，在第一子周期期间，每个窗口的处理包括在包括窗口中的前一个或多个网格级的第一子窗口、或者包括窗口中的后一个或多个网格级的第二子窗口内的前向递归和后向递归后向递归，且在第二子周期期间，处理元件中的每一个处理元件被配置为在包括窗口中的剩余网格级的第一和第二子窗口中的另一个内执行前向递归和后向递归。

[0207] 第11段、根据段落10所述的turbo解码器电路，其中，第一和第二子周期中的一个包括在周期中的时钟循环的半向下舍入(half rounding down)，第一和第二子周期中的另一个包括在周期中的时钟循环的剩余半向上舍入，并且在包括时钟循环的半向下舍入的第一子周期和第二子周期中的一个子周期期间，每个处理元件对包括窗口中的网格级的半向下舍入的第一或第二子窗口执行并行计算，并且在包括时钟循环的半向上舍入的第一子周期和第二子周期的另一个子周期期间，处理元件对包括窗口中的网格级的半向上舍入的第一或第二子窗口执行计算。

[0208] 第12段、根据段落11所述的turbo解码器电路，其中，处理元件被配置为在与子窗口内的完全前向递归和子窗口内的完全后向递归相关联的子周期内对子窗口执行计算，并且在执行完全前向递归和完全后向递归之后，任何剩余的时钟循环被处理元件用来执行与后续前向递归和后续后向递归的至少一部分相关联的计算。

[0209] 第13段、根据段落12所述的turbo解码器电路，其中，在第一子周期期间，上部解码器的处理元件被配置为执行与第一或第二子窗口中的相同的一个相关联的计算，并且下部解码器的处理元件被配置为执行与第一或第二子窗口中的另一个相关联的计算，并且

[0210] 在第二子周期期间，上部解码器的处理元件被配置为执行与在第一子周期期间未被处理元件处理的第一或第二子窗口相关联的计算，并且下部解码器的处理元件被配置为执行与在第一子周期期间未被处理元件处理的第一或第二子窗口中的另一个相关联的计算。

[0211] 第14段、根据段落10至13中任一段所述的turbo解码器电路，其中，前向递归根据执行与前向方向上的每个连续网格级相关联的计算的调度来生成与多个网格状态相对应的多个前向状态度量，并且后向递归根据执行与后向方向上的每个连续网格级相关联的计算的调度来生成与多个网格状态相对应的多个后向状态度量，并且前向递归根据用于前向递归的调度将前向状态度量存储在存储器中，或者后向递归根据用于后向递归的调度将后向状态度量存储在存储器中，并且前向递归或后向递归中的另一个从存储器加载存储的前向或后向状态度量，并组合前向和后向状态度量以根据用于前向或后向递归的调度来计算非本征软判决值。

[0212] 第15段、根据段落10至14中任一段所述的turbo解码器电路，其中，由处理元件根据前向递归和后向递归而执行的计算包括：接收与相邻网格级有关的前向或后向状态度量，将前向或后向状态度量与数据符号的先验、奇偶校验和系统软判决值进行组合，以及生成与另一相邻网格级有关的前向或后向状态度量，其中所接收的前向或后向状态度量在与先验、奇偶校验和系统软判决值组合之前被归一化。

[0213] 第16段、根据段落15所述的turbo解码器电路，其中，处理元件被配置为根据两步流水线来生成非本征软判决值，该两步流水线包括第一步，该第一步将前向和后向状态度量彼此组合并与奇偶校验软判决值组合以形成中间变量，以及第二步，该第二步将中间变量彼此组合、缩放中间变量的组合并将缩放的中间变量的组合与系统软判决值组合，并且流水线的两步在两个连续的时钟循环期间被执行，并且由流水线的步骤施加的延迟被包容在由可配置网络的预定调度施加的延迟中以避免竞争。

[0214] 第17段、根据段落1至16中任一段所述的turbo解码器电路，其中，帧中的数据符号的数量是可变的，并且由上部和下部解码器执行的计算的每个窗口的网格级的数量是相对于帧长度和上部和下部解码器的处理元件的数量来确定的。

[0215] 第18段、一种从接收到的信号中恢复数据符号的帧的turbo解码方法，该数据符号的帧包括该帧的每个数据符号的一个或多个奇偶校验和/或系统软判决值，该帧的数据符号已经用turbo编码器编码，该turbo编码器包括每个能够由网格表示的上部卷积编码器和下部卷积编码器、以及交织已经在上部卷积编码器和下部卷积编码器之间交织的编码数据的交织器，该方法包括：

[0216] 使用包括与上部卷积编码器相关联的多个上部处理元件的上部解码器，通过如下步骤来执行前向和后向迭代递归处理：

[0217] 在一系列连续时钟循环期间，在上部解码器的处理元件中的每一个处理元件处从可配置网络电路迭代地接收与数据符号有关的先验软判决值，其中该先验软判决值与表示上部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联，

[0218] 由处理单元中的每一个处理元件使用先验软判决值执行与该窗口相关联的并行计算，以便生成与数据符号有关的对应非本征软判决值，上部解码器的至少一个处理元件相对上部解码器的至少一个其他处理元件执行与不同数量的网格级相关联的窗口的计算，

[0219] 向可配置网络电路提供非本征软判决值，以及

[0220] 使用包括与下部卷积编码器相关联的多个下部处理元件的下部解码器，通过如下步骤来执行前向和后向迭代递归处理：

[0221] 在该系列连续时钟循环期间，在下部解码器的处理元件中的每一个处理元件处从可配置网络电路迭代地接收与数据符号有关的先验软判决值，其中该先验软判决值与表示下部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联，

[0222] 由处理单元中的每一个处理元件使用先验软判决值执行与该窗口相关联的并行计算，以便生成与数据符号有关的对应非本征软判决值，下部解码器的至少一个处理元件相对下部解码器的至少一个其他处理元件执行与不同数量的网格级相关联的窗口的计算，

[0223] 向可配置网络电路提供非本征软判决值，

[0224] 在连续时钟循环期间，迭代地控制可配置网络电路的配置，以通过交织由下部解码器提供的非本征软判决值来为上部解码器提供先验软判决值，并且通过交织由上部解码器提供的非本征软判决值来为下部解码器提供先验软判决值，由网络控制器控制的可配置网络电路执行的交织是根据预定调度的，这在一个或多个连续时钟循环的不同循环提供先验软判决值，以避免在相同时钟循环期间向上部或下部解码器的相同处理元件提供不同先验软判决值之间的竞争。

[0225] 第19段、一种接收器，用于检测和恢复已经用turbo码编码的数据符号的帧，该接

收器包括：

[0226] 检测电路，用于检测携带该数据符号的帧的接收到的信号，每个数据符号的帧包括帧的每个数据符号的一个或多个奇偶校验和/或系统软判决值，每个帧的数据符号已经用turbo编码器编码，该turbo编码器包括每个能够由网格表示的上部卷积编码器和下部卷积编码器，以及交织已经在上部卷积编码器和下部卷积编码器之间交织的编码数据的交织器，以及

[0227] turbo解码器电路，用于执行turbo解码处理以从接收到的信号中恢复数据符号的帧中的每一个，该turbo解码器电路包括：

[0228] 时钟，

[0229] 可配置网络电路，被配置为交织软判决值，

[0230] 上部解码器，包括与上部卷积编码器相关联的多个上部处理元件，上部解码器的处理元件中的每一个处理元件被配置为，在一系列连续时钟循环期间，从可配置网络电路迭代地接收与数据符号有关的先验软判决值，其中该先验软判决值与表示上部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联，以使用先验软判决值执行与该窗口相关联的并行计算以便生成与数据符号有关的对应非本征软判决值，以及被配置为向可配置网络电路提供非本征软判决值，上部解码器的至少一个处理元件被配置为相对上部解码器的至少一个其他处理元件执行与不同数量的网格级相关联的窗口的计算，以及

[0231] 下部解码器，包括与下部卷积编码器相关联的多个下部处理元件，下部解码器的处理元件中的每一个处理元件被配置为，在系列连续时钟循环期间，从可配置网络电路迭代地接收与数据符号有关的先验软判决值，其中该先验软判决值与表示下部卷积编码器的状态之间的可能路径的整数个连续网格级的窗口相关联，以使用先验软判决值执行与该窗口相关联的并行计算以便生成与数据符号有关的对应非本征软判决值，以及被配置为向可配置网络电路提供非本征软判决值，下部解码器的至少一个处理元件被配置为相对下部解码器的至少一个其他处理元件执行与不同数量的网格级相关联的窗口的计算，

[0232] 其中可配置网络电路包括网络控制器电路，网络控制器电路在连续时钟循环期间迭代地控制可配置网络电路的配置，以通过交织由下部解码器提供的非本征软判决值来为上部解码器提供先验软判决值，以及通过交织由上部解码器提供的非本征软判决值来为下部解码器提供先验软判决值，由网络控制器控制的可配置网络电路执行的交织是根据预定调度的，这在一个或多个连续时钟循环的不同循环提供先验软判决值，以避免在相同时钟循环期间向上部或下部解码器的相同处理元件提供不同先验软判决值之间的竞争。

[0233] 第20段、根据段落19所述的接收器，其中，每个帧中的数据符号的数量从一个到另一个动态地变化。

[0234] 第21段、一种形成无线通信网络的无线电接入网络的一部分的基础设施装备，该基础设施装备包括根据段落19所述的接收器。

[0235] 第22段、一种用于利用无线通信网络发送或接收数据的通信设备，该通信设备包括根据段落19所述的接收器。

[0236] 引用文献

[0237] [1]ETSI TS36.212 v 10.8.0(2013-06)LTE;Evolved Universal Terrestrial Radio Access(E-UTRA);Multiplexing and Channel Coding,V10.2.0ed.,2011.

[0238] [2]IEEE 802.16-2012 Standard for Local and Metropolitan Area Networks-Part 16:Air Interface for Broadband Wireless Access Systems,2012.

[0239] [3]C.Berrou,A.Glavieux,and P.Thitimajshima,“Near Shannon limit error-correcting coding and decoding:Turbo-codes(1),”in Proc.IEEE Int.Conf.on Communications,vol.2,Geneva,Switzerland,May 1993,pp.1064-1070.

[0240] [4]P.Robertson,E.Villebrun,and P.Hoeher,“A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain,”in Proc.IEEE Int.Conf.on Communications,vol.2,Seattle,WA,USA,June 1995,pp.1009-1013.

[0241] [5]IEEE 802.11n-2009 Standard for Information Technology-Telecommunications and Information Exchange between Systems-Local and Metropolitan Area Networks-Specific Requirements-Part 11:Wireless LAN Medium Access Control(MAC)and Physical Layer(PHY),2009.

[0242] [6]D.J.C.MacKay and R.M.Neal,“Near Shannon limit performance of low density parity check codes,”Electron.Lett.,vol.32,no.18,pp.457-458,Aug.1996.

[0243] [7]M.Fossorier,“Reduced complexity decoding of low-density parity check codes based on belief propagation,”IEEE Trans.Commun.,vol.47,no.5,pp.673-680,May 1999.

[0244] [8]5G Radio Access.Ericsson White Paper,June 2013.

[0245] [9]V.A.Chandrasetty and S.M.Aziz,“FPGA implementation of a LDPC decoder using a reduced complexity message passing algorithm,”Journal of Networks,vol.6,no.1,pp.36-45,Jan.2011.

[0246] [10]T.Ilnseher,F.Kienle,C.Weis,and N.Wehn,“A 2.15GBit/s turbo code decoder for LTE Advanced base station applications,”in Proc.Int.Symp.on Turbo Codes and Iterative Information Processing,Gothenburg,Sweden,Aug.2012,pp.21-25.

[0247] [11]L.Fanucci,P.Ciao,and G.Colavolpe,“VLSI design of a fully-parallel high-throughput decoder for turbo gallager codes,”IEICE Trans.Fundamentals,vol.E89-A,no.7,pp.1976-1986,July 2006.

[0248] [12]D.Vogrig,A.Gerosa,A.Neviani,A.Graell I Amat,G.Montorsi,and S.Benedetto,“A 0.35-μm CMOS analog turbo decoder for the 40-bit rate 1/3 UMTS channel code,”IEEE J.Solid-State Circuits,vol.40,no.3,pp.753-762,2005.

[0249] [13]Q.T.Dong,M.Arzel,C.J.Jego,and W.J.Gross,“Stochastic decoding of turbo codes.”IEEE Trans.Signal Processing,vol.58,no.12,pp.6421-6425,Dec.2010.

[0250] [14]A.Nimbalker,Y.Blankenship,B.Classon,and T.K.Blankenship,“ARP and QPP interleavers for LTE turbo coding,”in Proc.IEEE Wireless Commun.Networking Conf.,Las Vegas,NV,USA,Mar.2008,pp.1032-1037.

[0251] [15]L.Li,R.G.Maunder,B.M.Al-Hashimi,and L.Hanzo,“A low-complexity turbo decoder architecture for energy-efficient wireless sensor networks,”IEEE Trans.VLSI Syst.,vol.21,no.1,pp.14-22,Jan.2013.[Online].Available:

http://eprints.soton.ac.uk/271820/

[0252] [16]P.Radosavljevic,A.de Baynast,and J.R.Cavallaro,“Optimized message passing schedules for LDPC decoding,”in Asilomar Conf.Signals Systems and Computers,no.1,Pacific Grove,CA,USA,Oct.2005,pp.591-595.

[0253] [17]CN 102611464

[0254] [18]CN 102723958

[0255] [19]WO 2011/082509

[0256] [20]“A 122Mb/s Turbo decoder using a mid-range GPU”by Xianjun J.,et al,published at Wireless Communications and Mobile Computing Conference (IWCMC),2013 9th International,pages 1090-1094,1-5 July 2013.

[0257] [21]“Turbo IP Core User Guide”,UG-Turbo,2015.11.11 Altera.

[0258] [22]L.F.Gonzalez-Perez,L.C.Yllescas-Calderon,and R.Parra-Michel,“Parallel and Configurable Turbo Decoder Implementation for 3GPP-LTE,”in 2013 Int.Conf.Reconfigurable Comput.FPGAs,pp.1-6,IEEE,Dec 2013.

[0259] [23]Yufei Wu,B.D.Woerner and T.K.Blankenship,"Data width requirements in SISO decoding with module normalization,"in IEEE Transactions on Communications,vol.49,no.11,pp.1861-1868,Nov 2001.

[0260] [24]J.Zhang and M.P.C.Fossorier,“Shuffled iterative decoding,”IEEE Trans.Commun.,vol.53,no.2,pp.209-213,Feb.2005.

[0261] [25]C.Chang,“Arbitrary size Benes networks”,Parallel Processing Letters,vol.7,no.3,Sept 1997.

[0262] [26]PCT/EP2015/067527

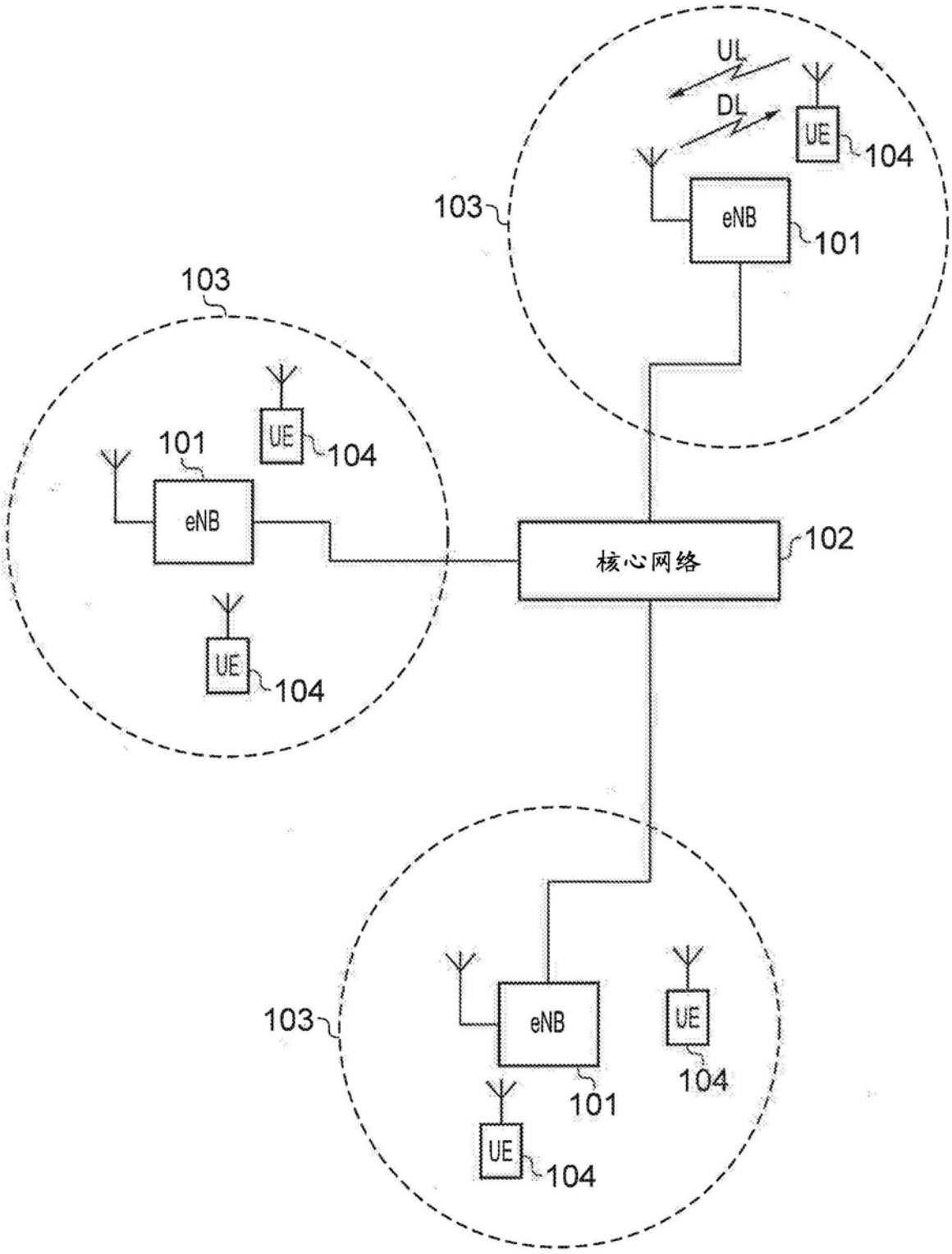

UL
DL
UE
104
103
eNB
101
103
101
UE
104
eNB
核心网络
102
UE
104
103
eNB
UE
101
104
UE
104

图1

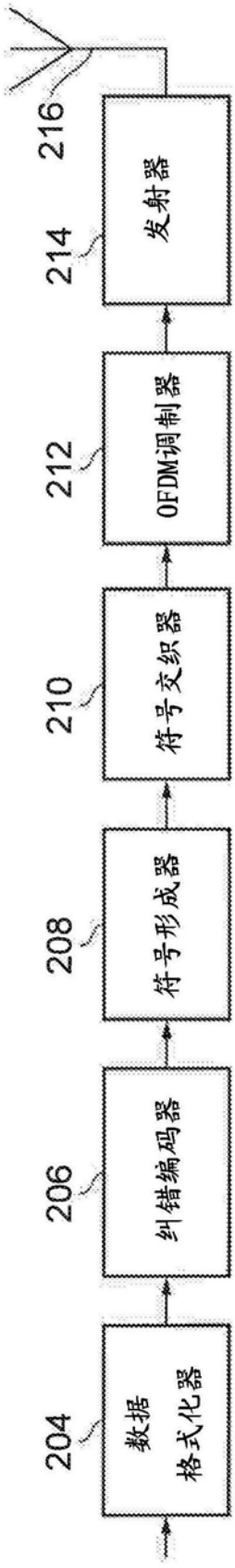

216
214
发射器
212
OFDM调制器
210
符号交织器
208
符号形成器
206
纠错编码器
204
数据
格式化器

图2

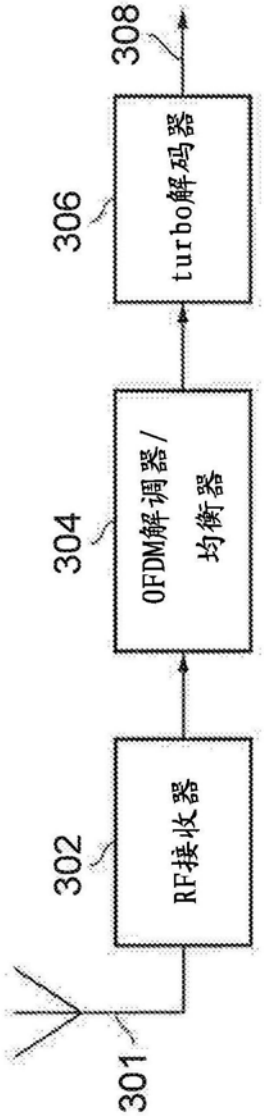


图3

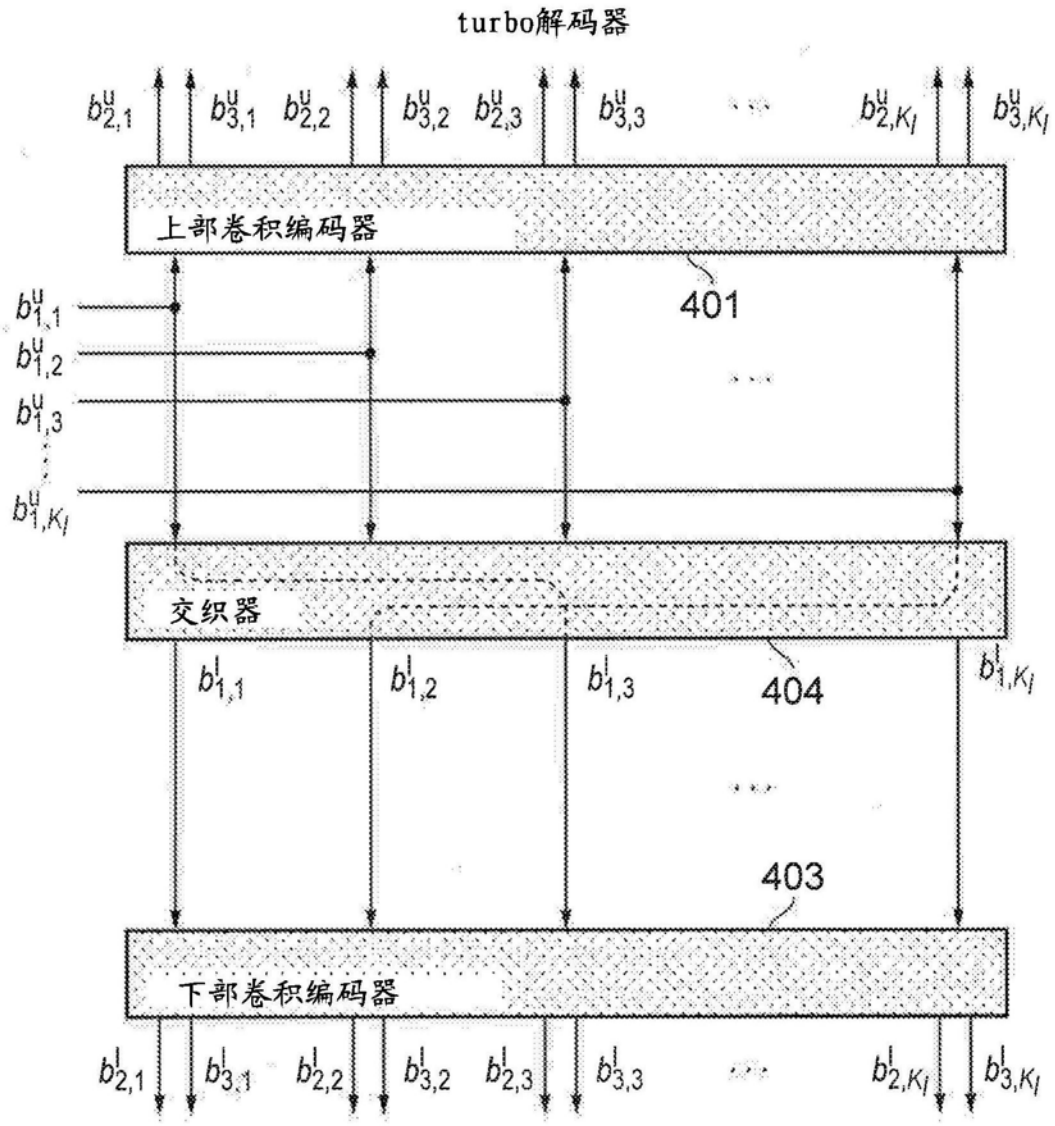


图4

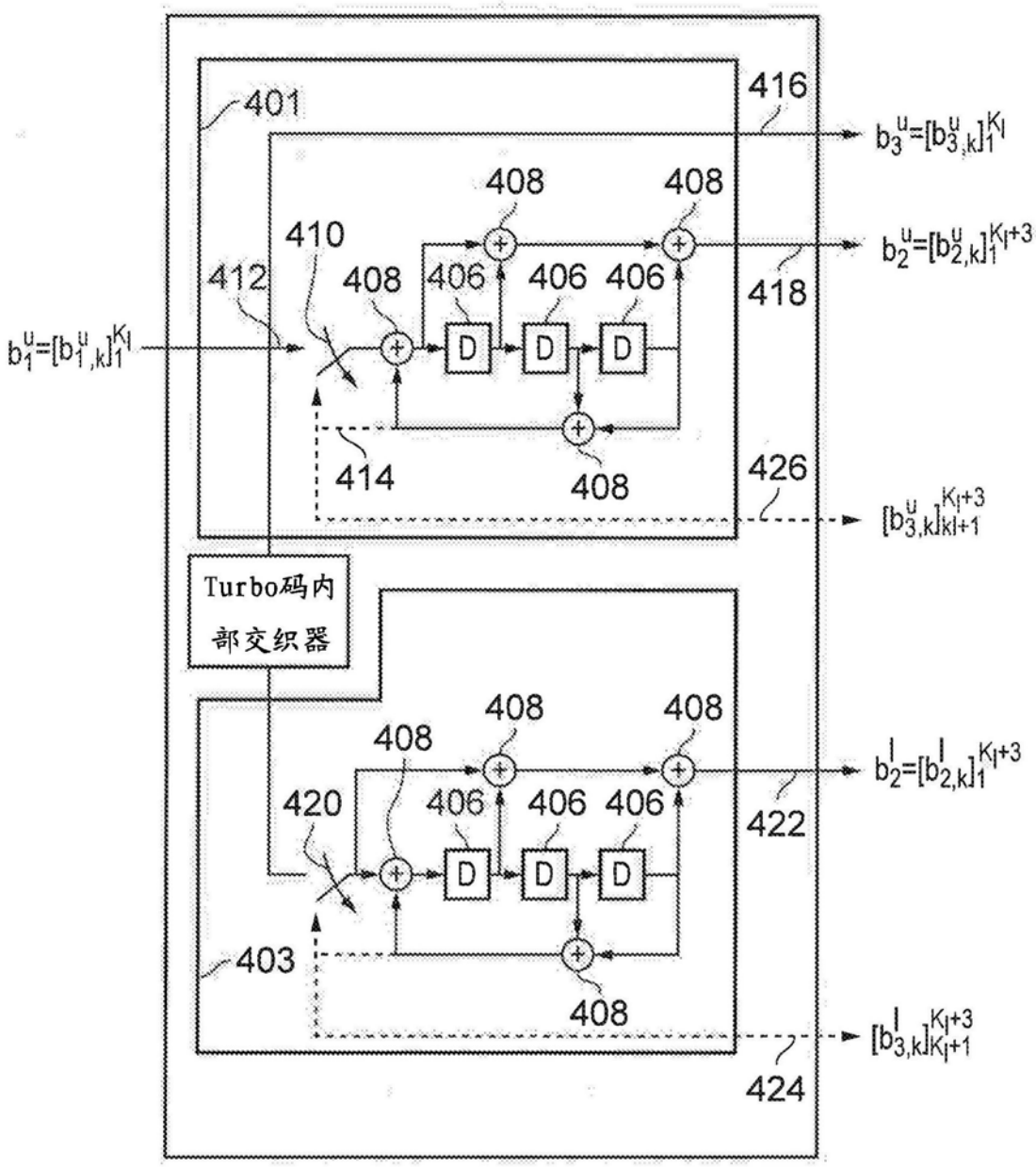

401
416
$b_3^u=[b_{3,k}^u]_1^{K_I}$
408
408
410
412
408
406
406
406
$b_2^u=[b_{2,k}^u]_1^{K_I+3}$
418
$b_1^u=[b_{1,k}^u]_1^{K_I}$
D
D
D
414
408
426
$[b_{3,k}^u]_{K_I+1}^{K_I+3}$
Turbo码内部交织器
408
408
408
$b_2^I=[b_{2,k}^I]_1^{K_I+3}$
420
406
406
406
422
D
D
D
403
408
$[b_{3,k}^I]_{K_I+1}^{K_I+3}$
424

图5

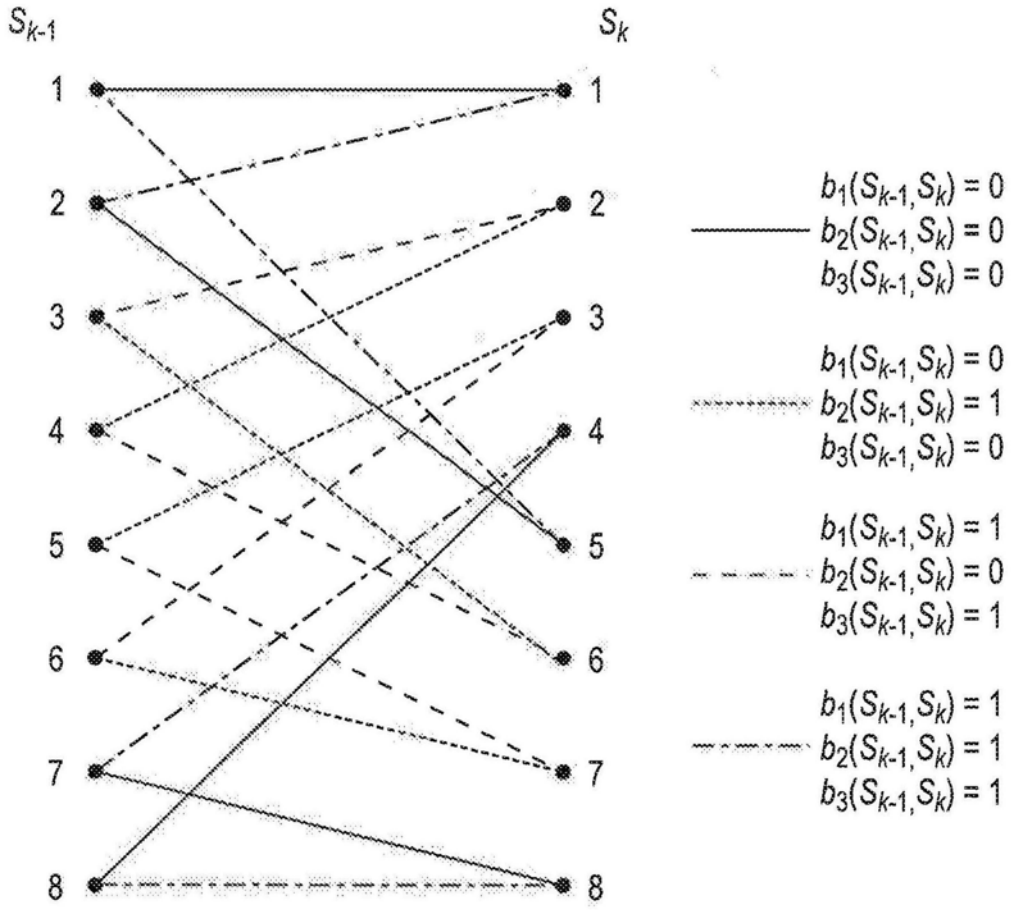

S_{k-1}
S_k
1
2
3
4
5
6
7
8
$b_1(S_{k-1},S_k) = 0$
$b_2(S_{k-1},S_k) = 0$
$b_3(S_{k-1},S_k) = 0$
$b_1(S_{k-1},S_k) = 0$
$b_2(S_{k-1},S_k) = 1$
$b_3(S_{k-1},S_k) = 0$
$b_1(S_{k-1},S_k) = 1$
$b_2(S_{k-1},S_k) = 0$
$b_3(S_{k-1},S_k) = 1$
$b_1(S_{k-1},S_k) = 1$
$b_2(S_{k-1},S_k) = 1$
$b_3(S_{k-1},S_k) = 1$

图6

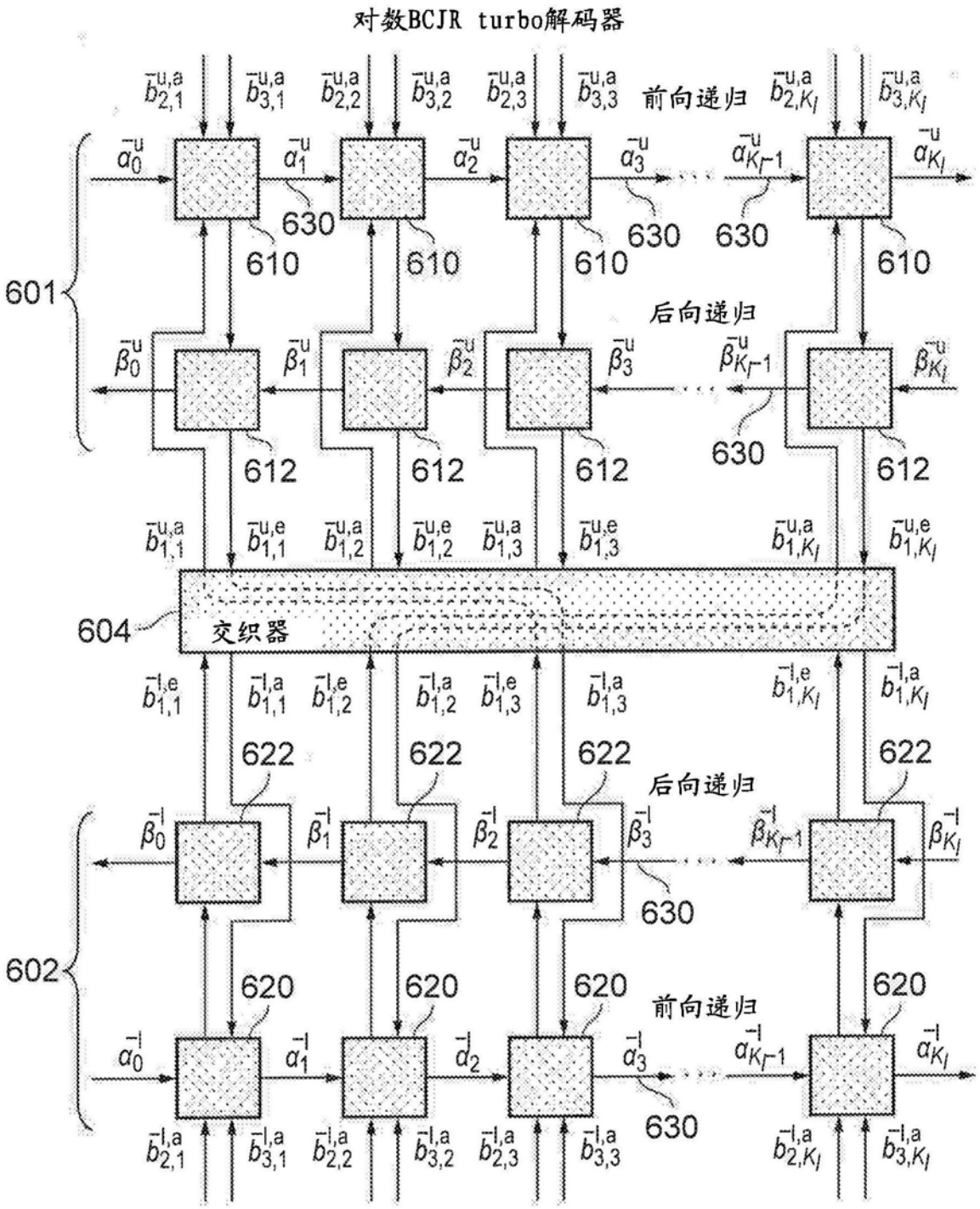

对数BCJR turbo解码器
前向递归
后向递归
后向递归
前向递归
交织器
601
602
604
610
612
620
622
630

图7

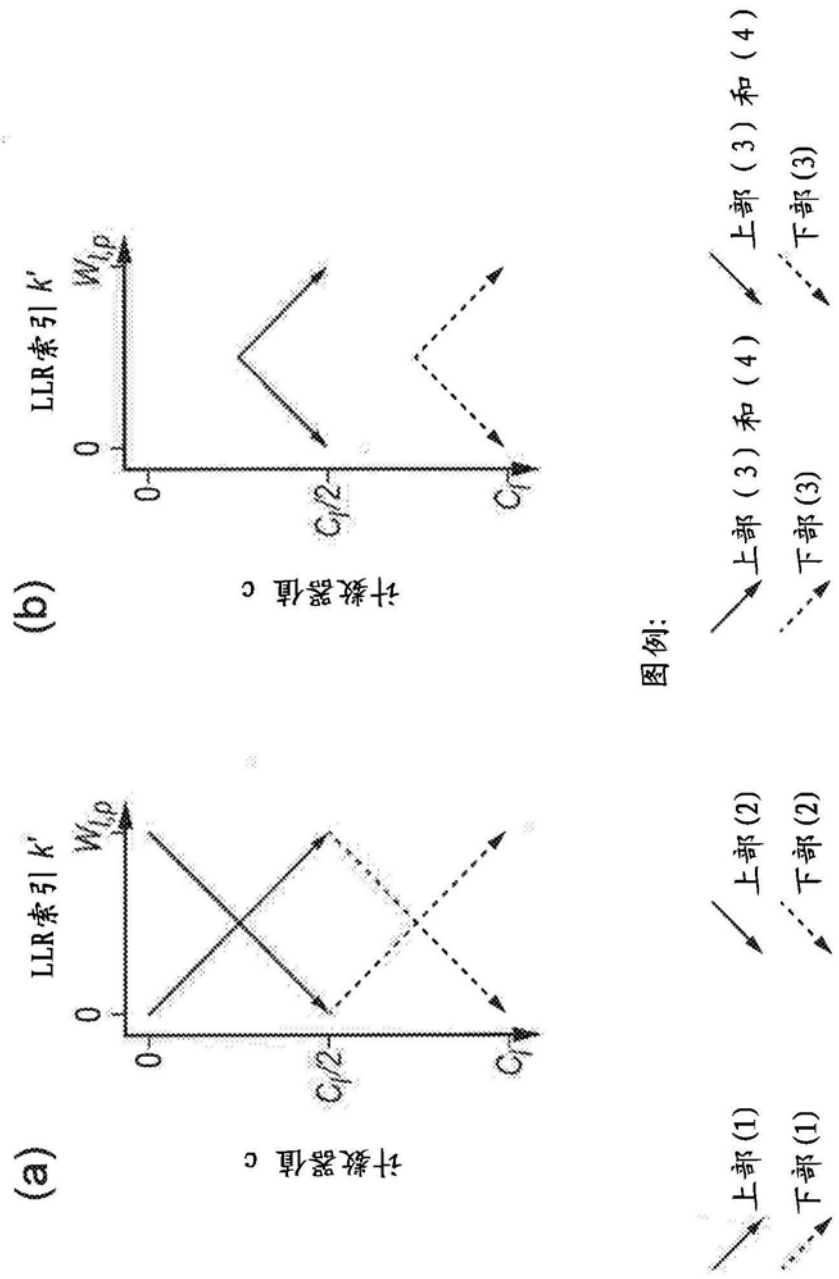

(a)
(b)
LLR索引 k'
W_{LLR}
0
计数器值 c
$C_I/2$
C_I
图例:
上部 (1)
下部 (1)
上部 (2)
下部 (2)
上部 (3) 和 (4)
下部 (3)
上部 (3) 和 (4)
下部 (3)

图8

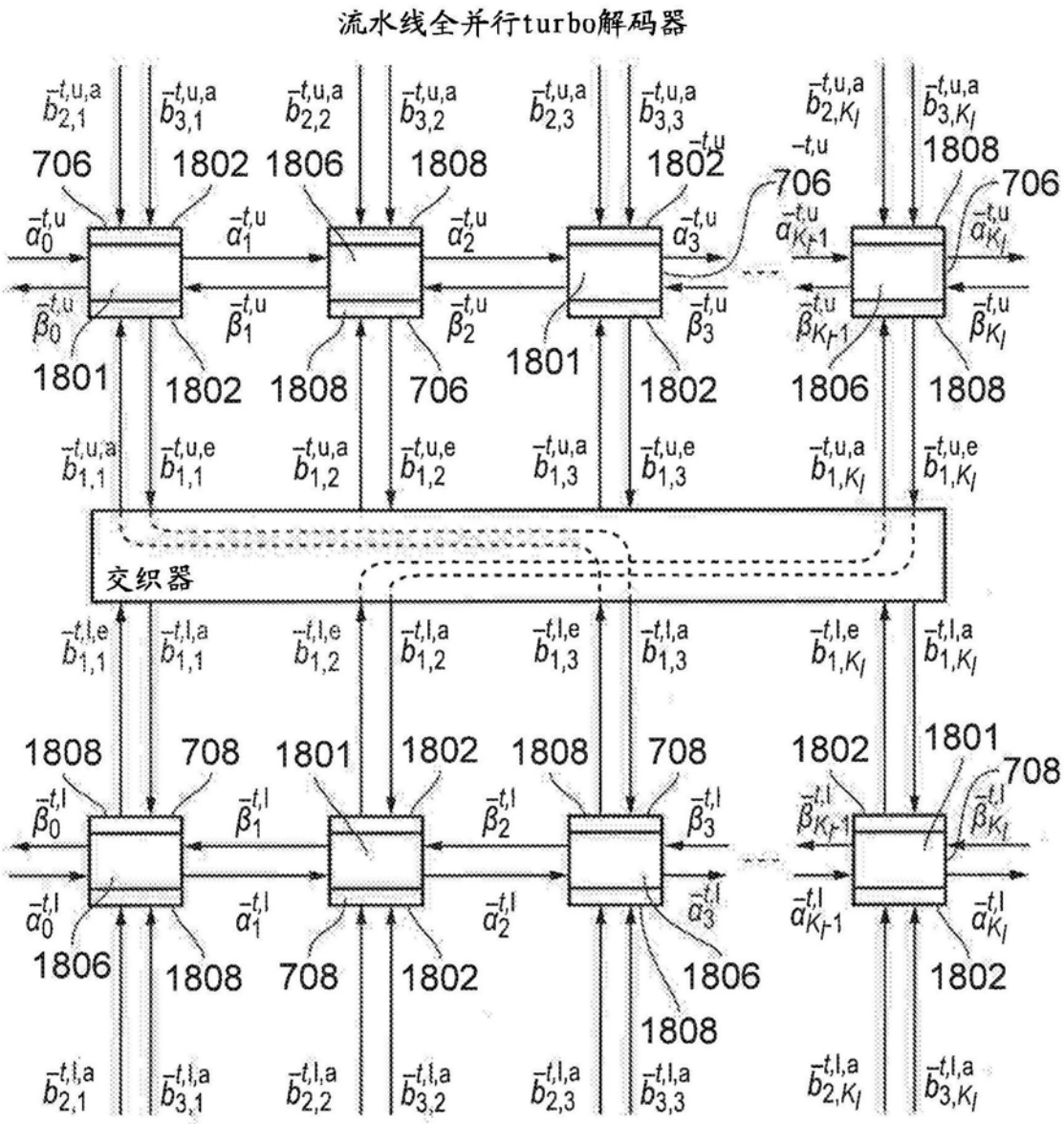

流水线全并行turbo解码器
706
1802
1806
1808
1802
706
1808
706
1801
1802
1808
706
1801
1802
1806
1808
交织器
1808
708
1801
1802
1808
708
1802
1801
708
1806
1808
708
1802
1808
1806
1802

图9

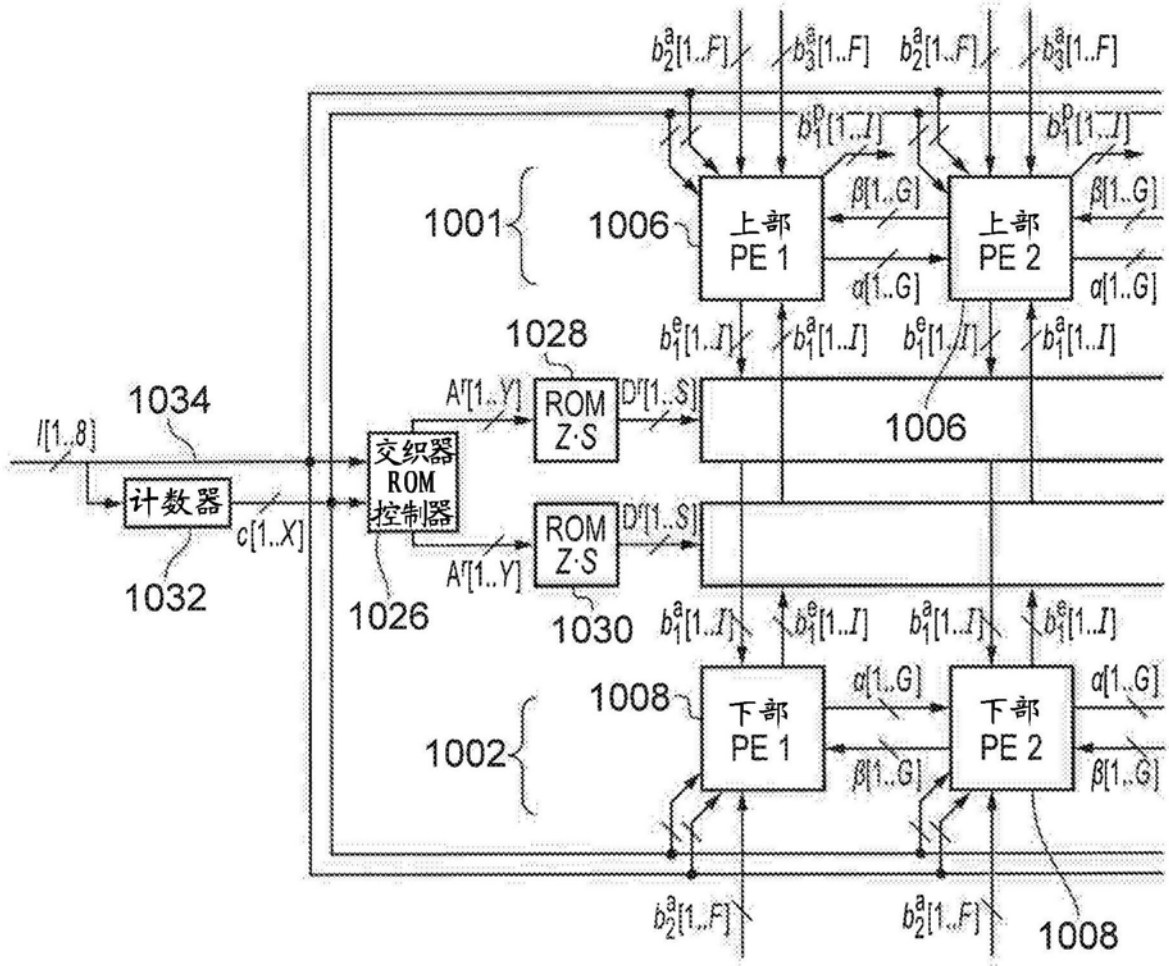

1001
1006
上部 PE 1
上部 PE 2
1028
ROM Z·S
交织器 ROM 控制器
计数器
1034
1032
1026
1030
1006
1008
下部 PE 1
下部 PE 2
1002
1008

图10

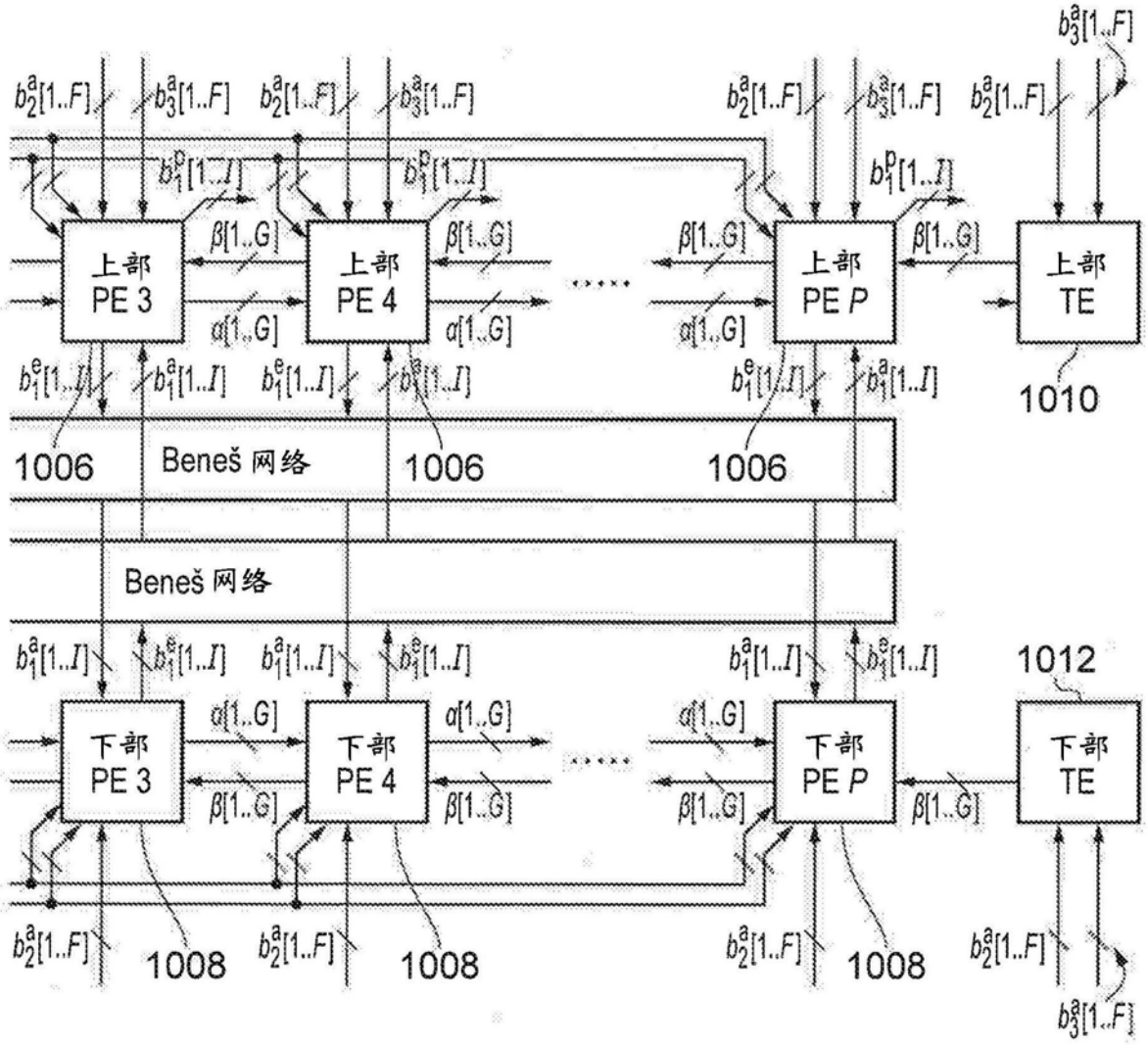

上部 PE 3
上部 PE 4
上部 PE P
上部 TE
Beneš 网络
Beneš 网络
下部 PE 3
下部 PE 4
下部 PE P
下部 TE
1006
1006
1006
1010
1012
1008
1008
1008

图10(续)

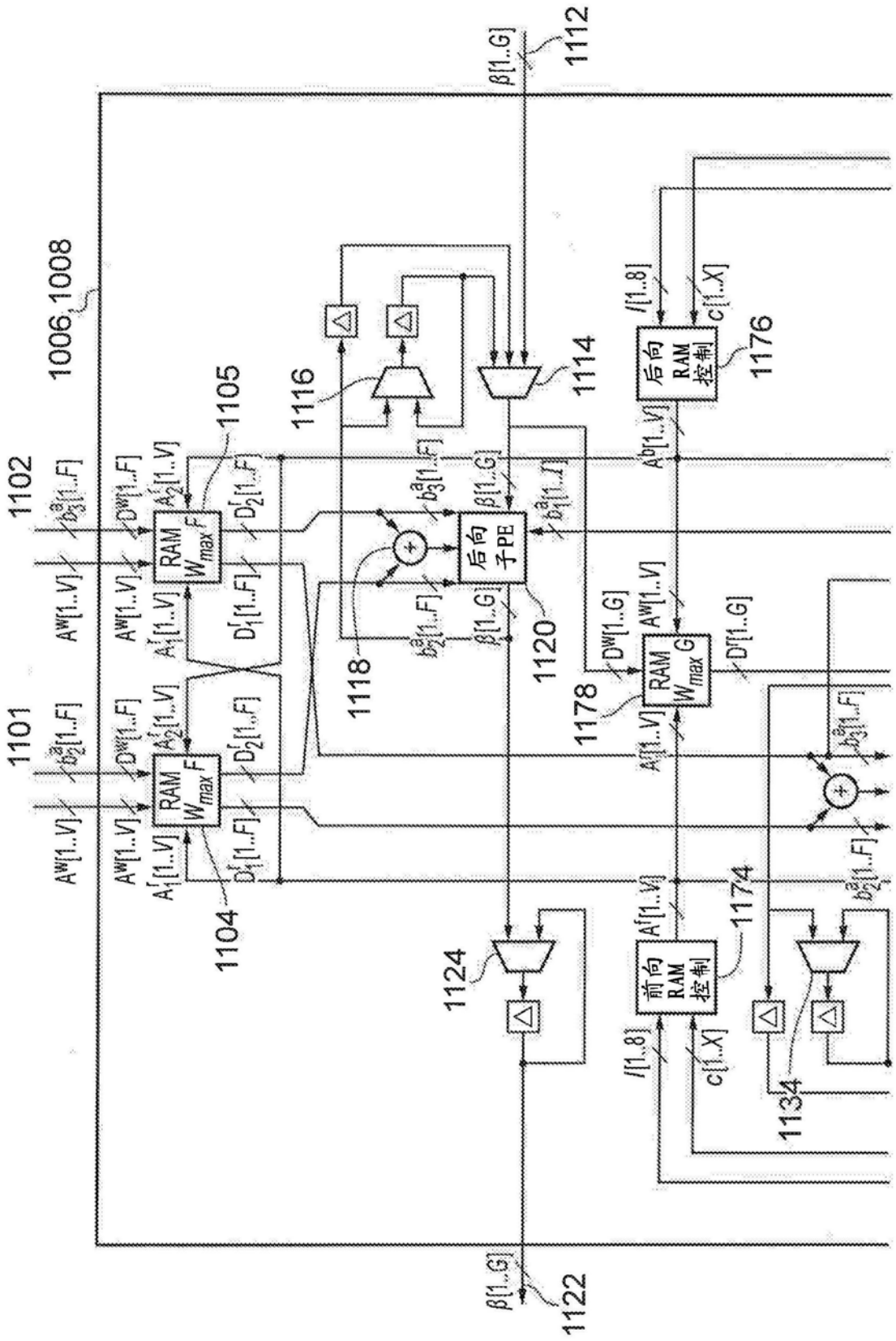

1112
β[1..G]
1006 1008
1102
1101
1105
1116
1114
1176
1118
1120
1178
1104
1124
1174
1134
1122
RAM
后向 子PE
后向 RAM 控制
前向 RAM 控制

图11

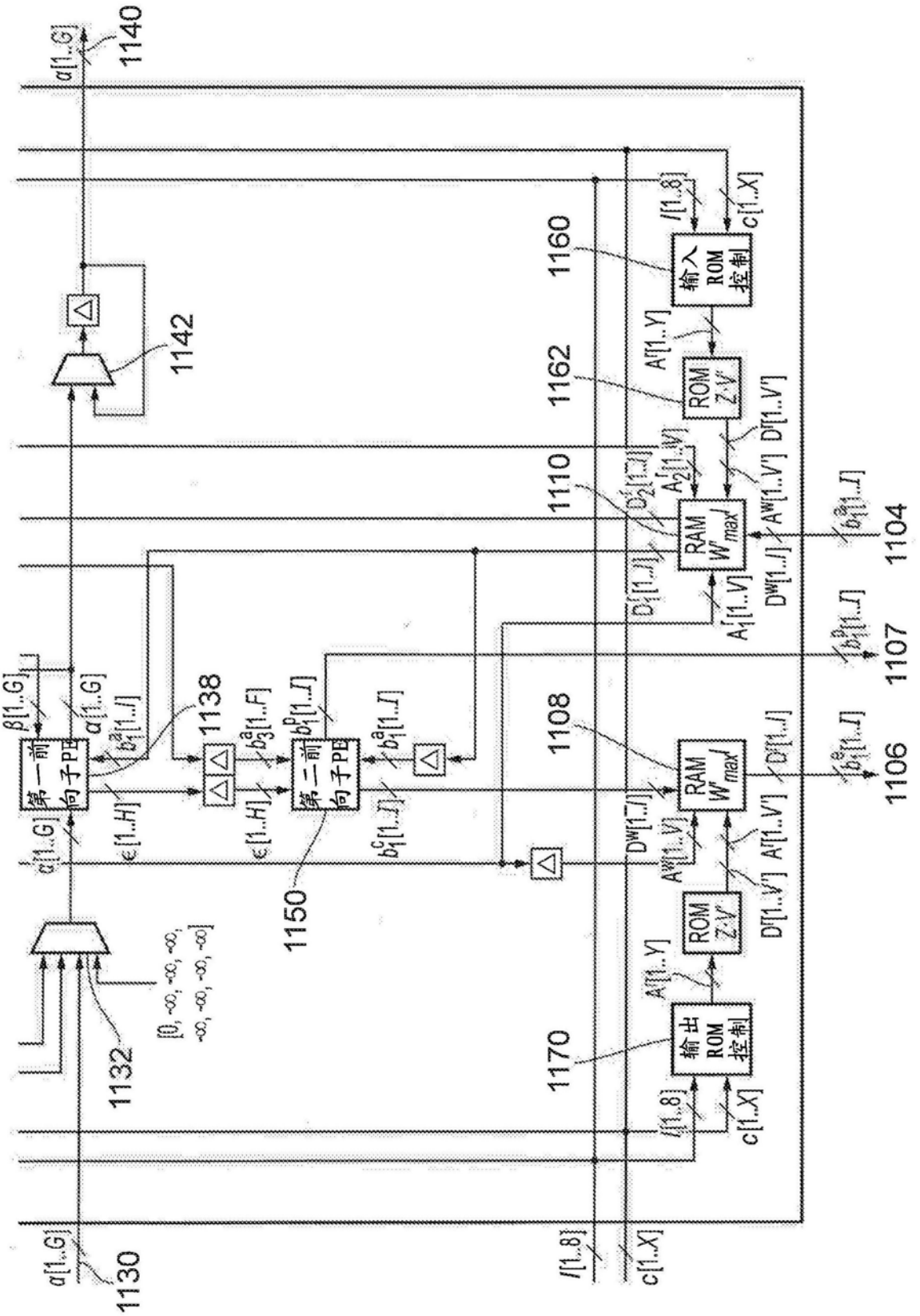

1140
1142
1160
1162
1110
1104
1107
1108
1106
1150
1138
1132
1170
1130
输入 ROM 控制
输出 ROM 控制
RAM
ROM
第一前向子PB
第二前向子PB

图11(续)

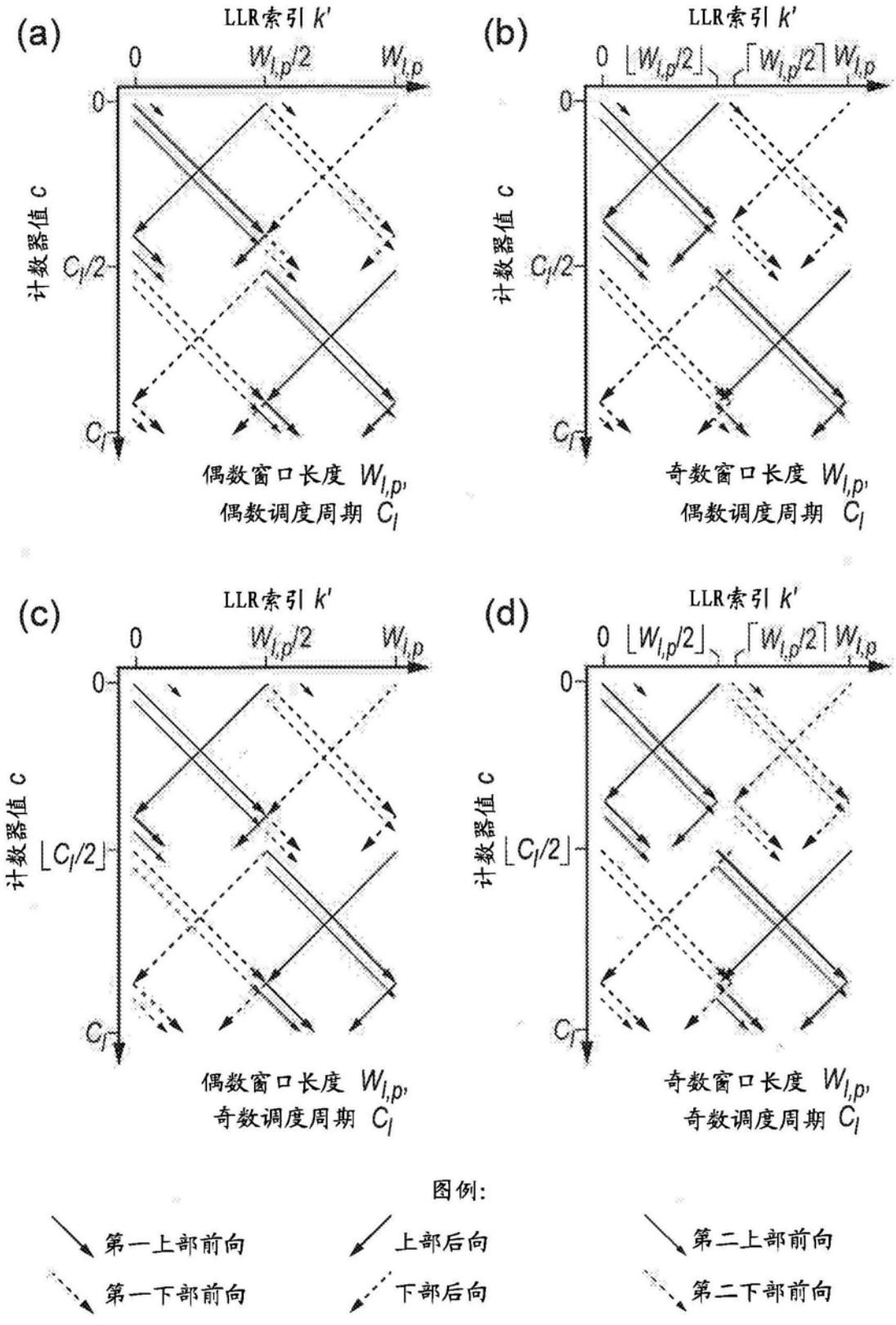

(a)
LLR索引 k'
0
$W_{l,p}/2$
$W_{l,p}$
计数器值 c
$C_l/2$
C_l
偶数窗口长度 $W_{l,p}$,
偶数调度周期 C_l
(b)
LLR索引 k'
$\lfloor W_{l,p}/2 \rfloor$
$\lceil W_{l,p}/2 \rceil$
奇数窗口长度 $W_{l,p}$,
偶数调度周期 C_l
(c)
$\lfloor C_l/2 \rfloor$
偶数窗口长度 $W_{l,p}$,
奇数调度周期 C_l
(d)
奇数窗口长度 $W_{l,p}$,
奇数调度周期 C_l
图例:
第一上部前向
上部后向
第二上部前向
第一下部前向
下部后向
第二下部前向

图12

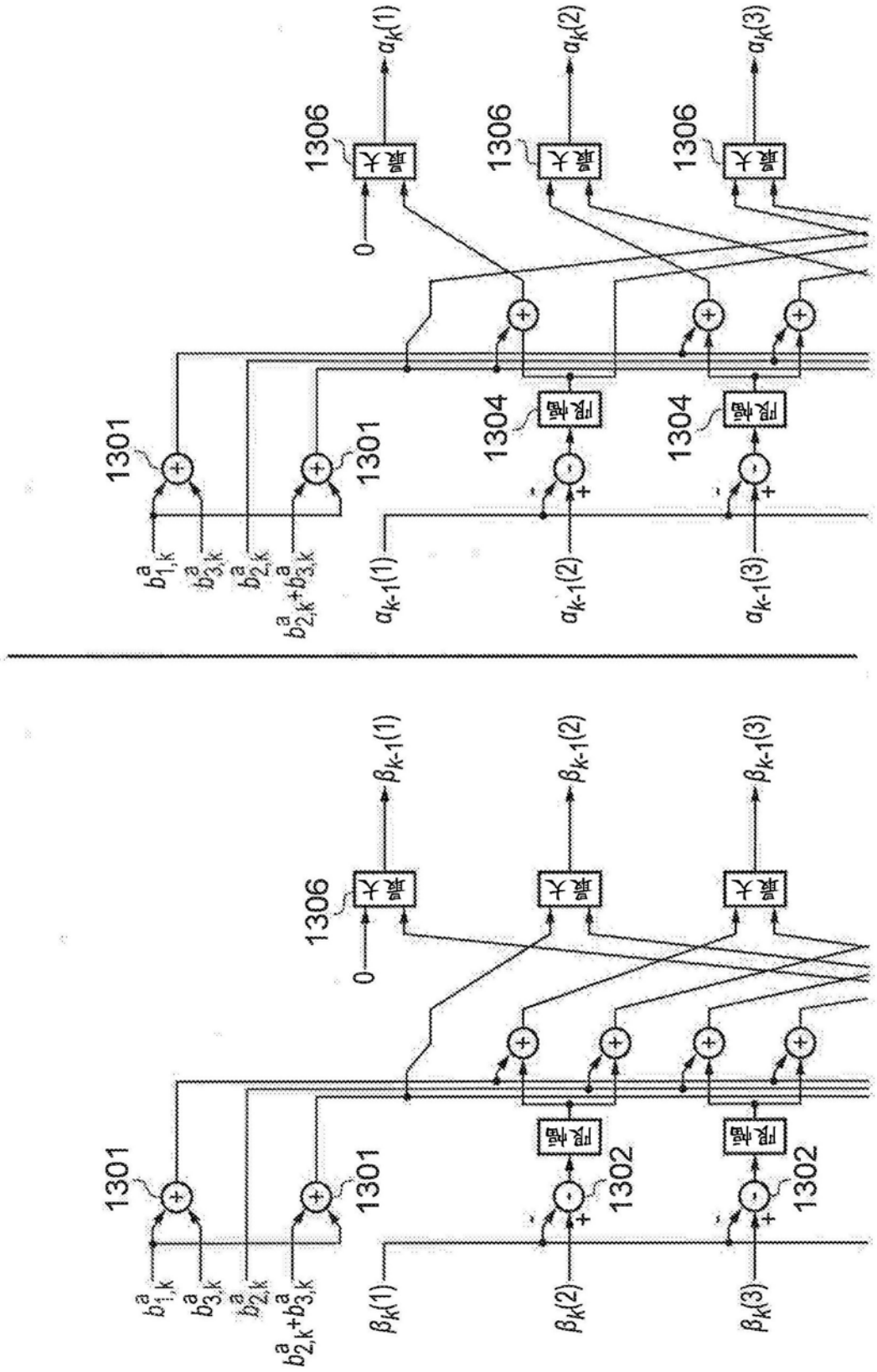

1306
1304
1301
1302
0
$a_k(1)$
$a_k(2)$
$a_k(3)$
$a_{k-1}(1)$
$a_{k-1}(2)$
$a_{k-1}(3)$
$\beta_k(1)$
$\beta_k(2)$
$\beta_k(3)$
$\beta_{k-1}(1)$
$\beta_{k-1}(2)$
$\beta_{k-1}(3)$
$b^a_{1,k}$
$b^a_{3,k}$
$b^a_{2,k}$
$b^a_{2,k}+b^a_{3,k}$

图13

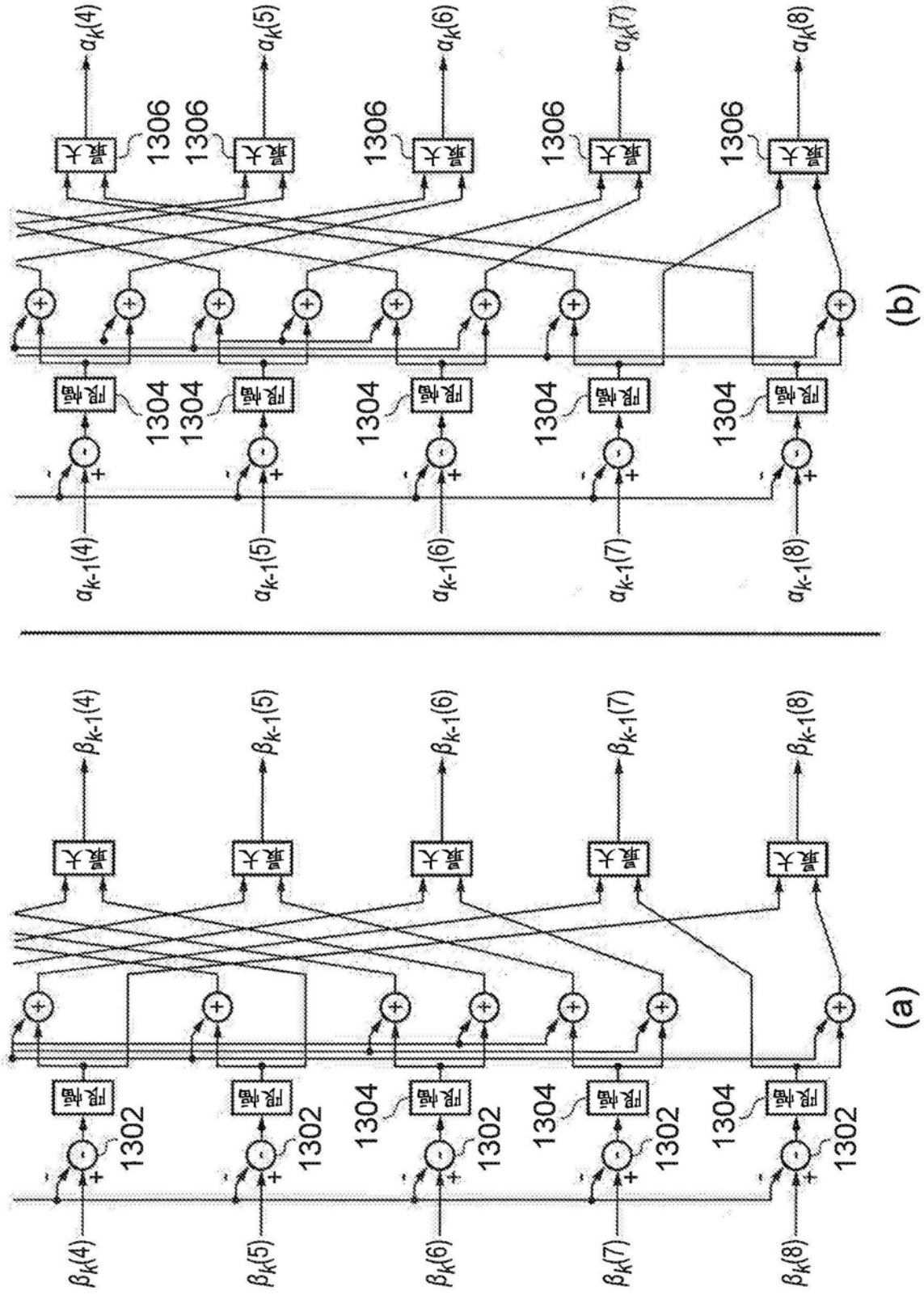


图13(续)

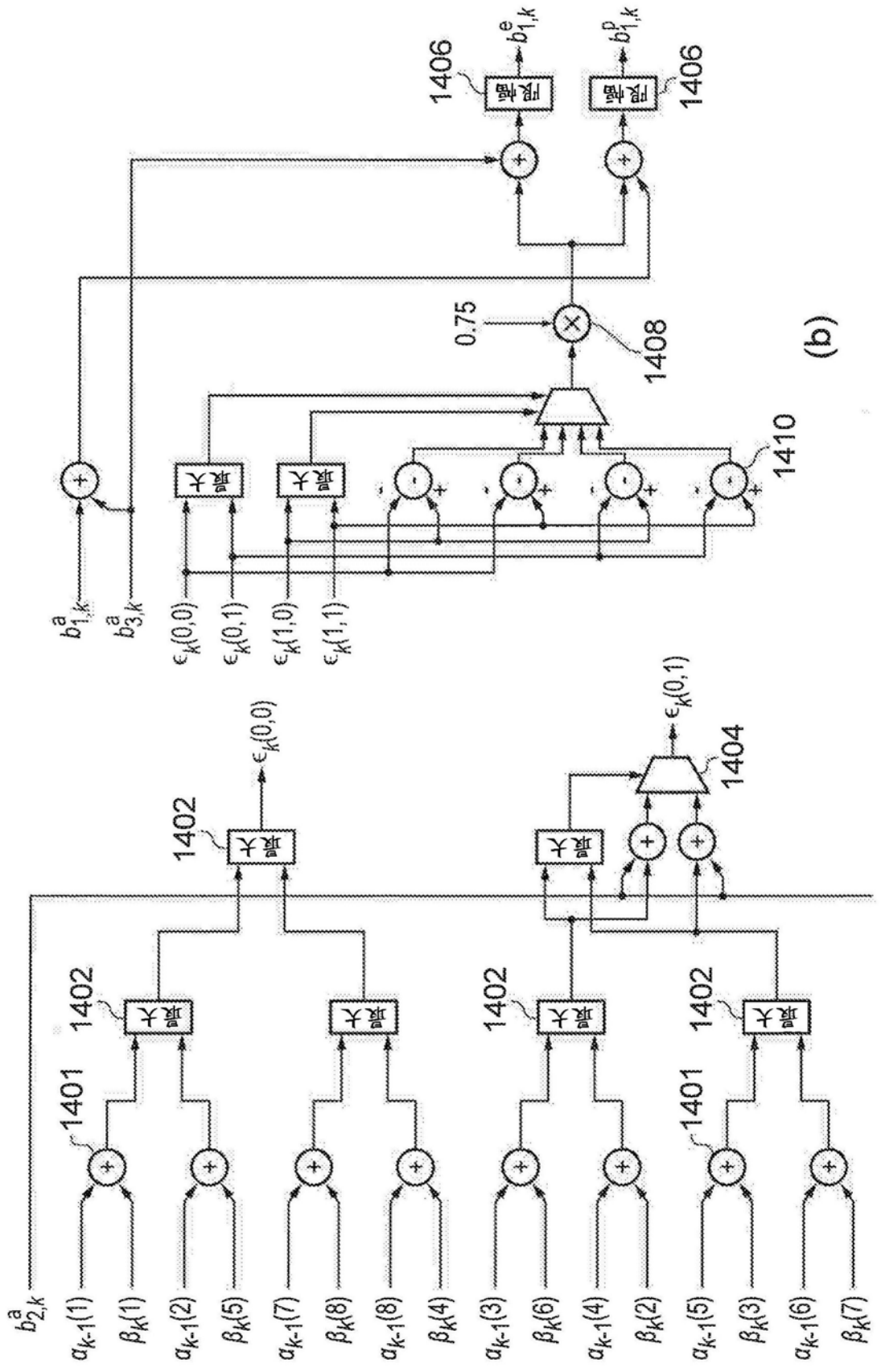


图14

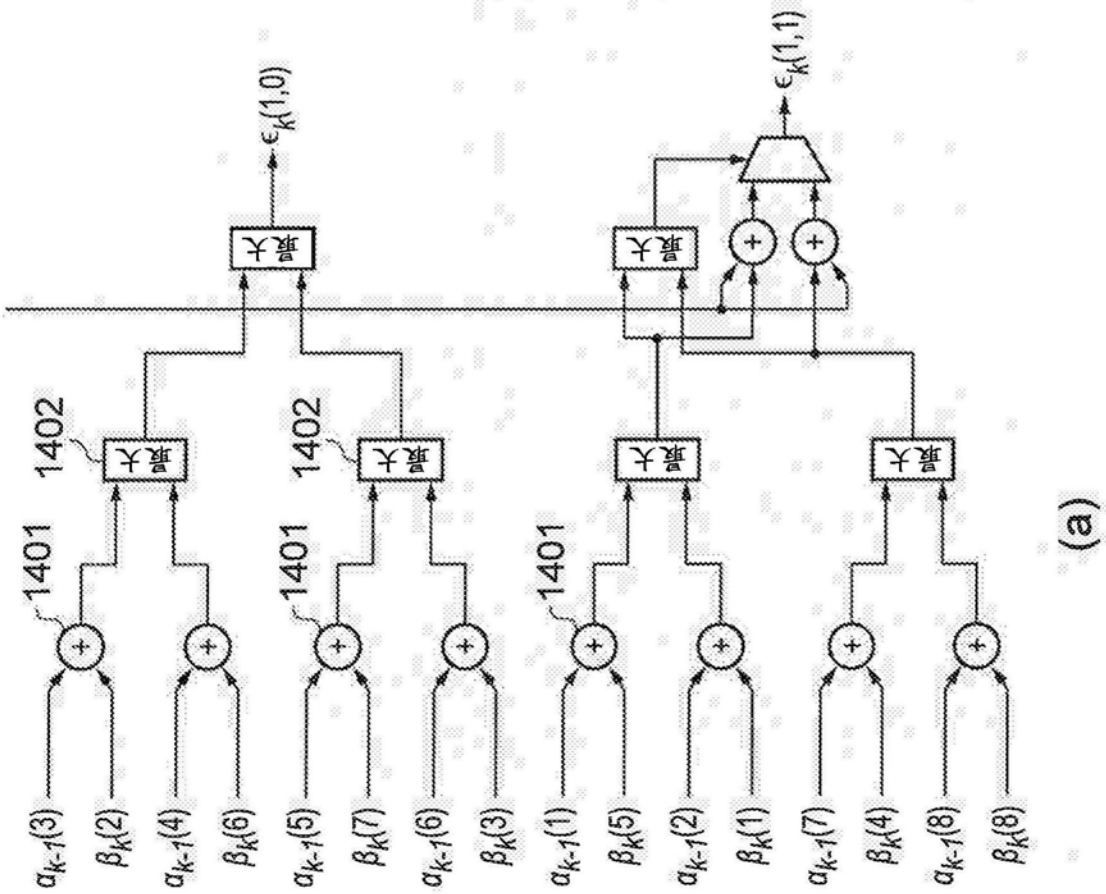

$\epsilon_k(1,0)$
$\epsilon_k(1,1)$
最大
1402
1401
$\alpha_{k-1}(3)$
$\beta_k(2)$
$\alpha_{k-1}(4)$
$\beta_k(6)$
$\alpha_{k-1}(5)$
$\beta_k(7)$
$\alpha_{k-1}(6)$
$\beta_k(3)$
$\alpha_{k-1}(1)$
$\beta_k(5)$
$\alpha_{k-1}(2)$
$\beta_k(1)$
$\alpha_{k-1}(7)$
$\beta_k(4)$
$\alpha_{k-1}(8)$
$\beta_k(8)$
(a)

图14(续)

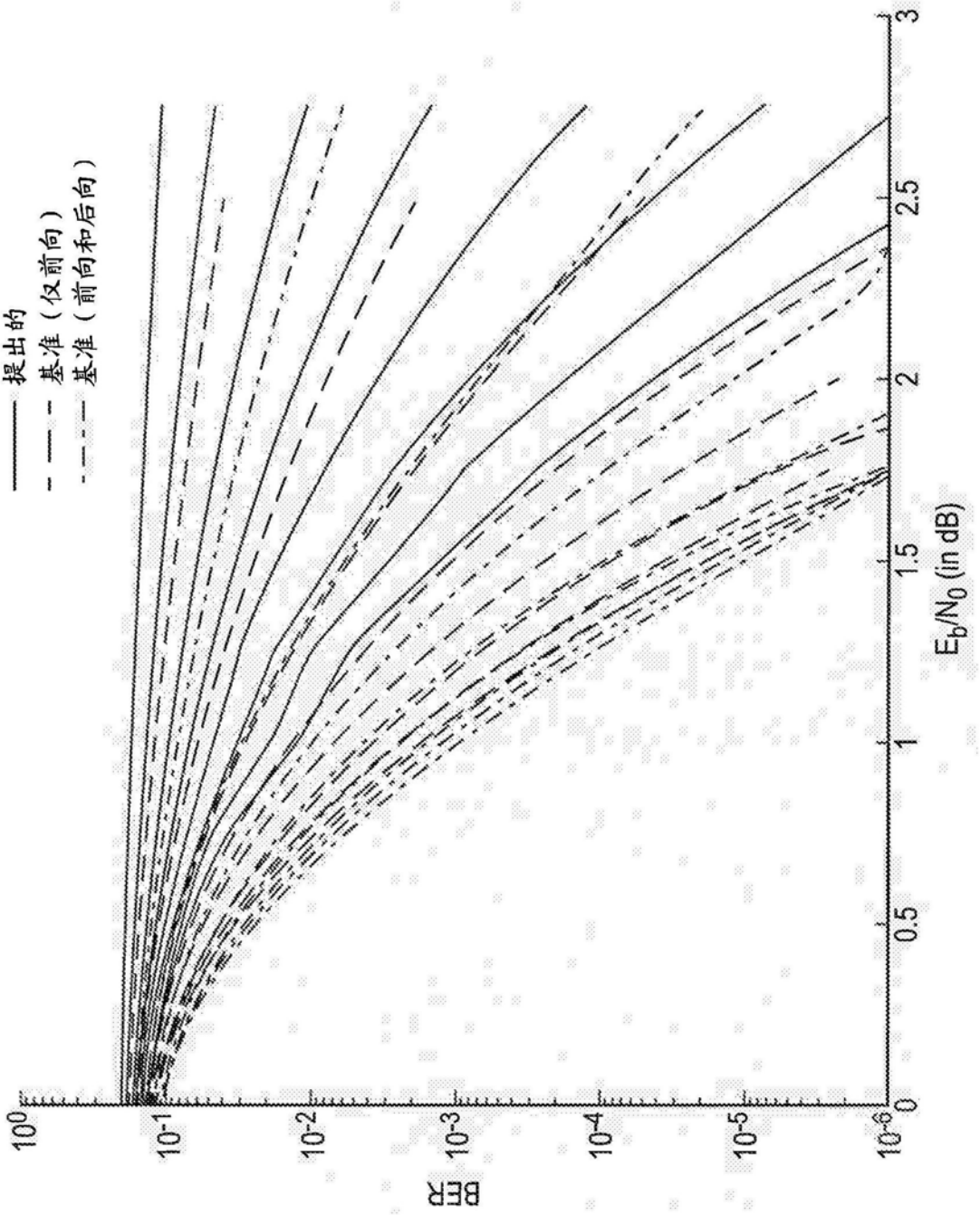

提出的
基准（仅前向）
基准（前向和后向）
BER
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0
0.5
1
1.5
2
2.5
3
E_b/N_0 (in dB)

图15

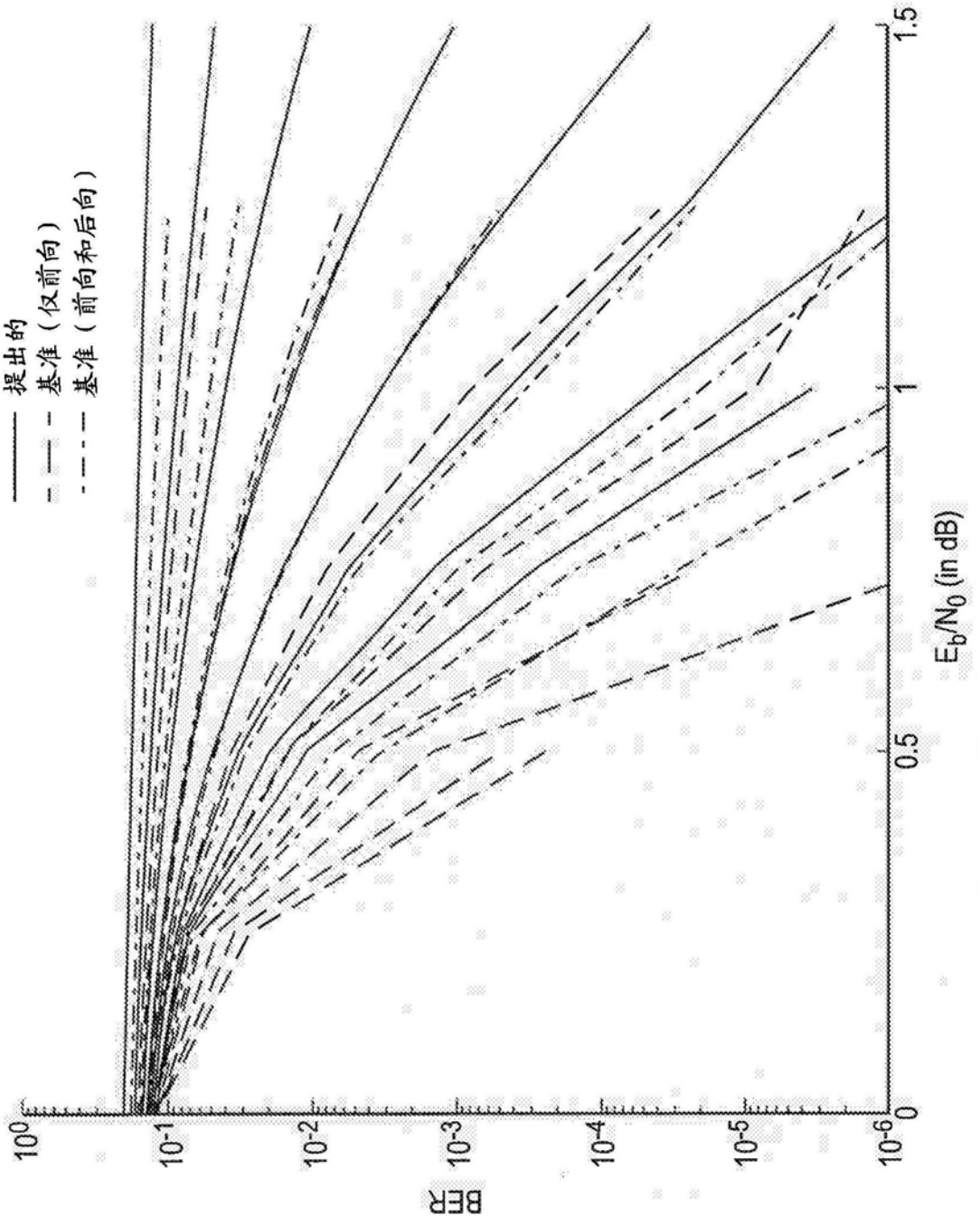

提出的
基准（仅前向）
基准（前向和后向）
BER
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0
0.5
1
1.5
E_b/N_0 (in dB)

图16