(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 105187185 A
(43)申请公布日 2015.12.23

(21)申请号 201510673207.X

(22)申请日 2005.12.22

(30)优先权数据
60/638,469 2004.12.22 US

(62)分案原申请数据
200580048263.9 2005.12.22

(71)申请人 高通股份有限公司
地址 美国加利福尼亚

(72)发明人 R·保兰基 A·汉德卡尔

(74)专利代理机构 永新专利商标代理有限公司 72002
代理人 戴开良 王英

(51)Int.Cl.
H04L 5/00(2006.01)
H04L 9/06(2006.01)
H04B 1/7143(2011.01)

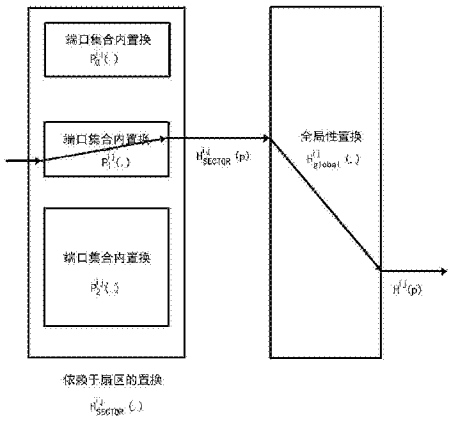
权利要求书2页 说明书12页 附图6页

(54)发明名称

用于在多址通信网络中进行灵活跳变的方法和装置

(57)摘要

本发明涉及一种在多址通信网络中进行灵活跳变的方法和装置。这种方法包括确定第一数量的子载波，确定第二数量的跳变端口，确定第三数量的种子，以及基于所述第一数量的子载波、所述第二数量的跳变端口和所述第三数量的种子产生至少一个跳变模式。这种技术能够用来有效地设计通信系统中的随机跳变模式，通过频繁地更新这些模式，为不同的小区/扇区产生不同的模式，并且为块跳变产生附近的频率子载波的模式。



CN 105187185 A

1. 一种在通信系统中用于利用至少一个处理器产生随机跳变模式的方法，所述方法包括：

确定第一数量的子载波；

确定第二数量的跳变端口；

确定第三数量的种子；以及

在所述通信系统中利用所述至少一个处理器产生能够用于信号传输的至少一个跳变模式，其中所述至少一个跳变模式是基于所述第一数量的子载波、所述第二数量的跳变端口和基于所述第三数量的种子的伪随机置换确定出来的，

其中产生所述至少一个跳变模式包括：

利用所述伪随机置换的至少一部分为所述第二数量的跳变端口的至少一个子集组产生第一跳变模式；以及

利用所述伪随机置换基于所述第一跳变模式产生第二跳变模式。

2. 如权利要求 1 所述的方法，其中所述第三数量的种子中的至少一个是基于系统时间加以确定的。

3. 如权利要求 1 所述的方法，其中所述第三数量的种子中的至少一个是基于扇区 ID 加以确定的。

4. 如权利要求 1 所述的方法，其中所述第三数量的种子中的至少一个是基于小区 ID 加以确定的。

5. 如权利要求 1 所述的方法，其中所述第一数量大于所述第二数量。

6. 如权利要求 1 所述的方法，还包括频繁地改变所述至少一个跳变模式。

7. 如权利要求 1 所述的方法，还包括：

将所述第二数量的跳变端口分成多个较小的跳变端口子集组；以及

为所述多个较小的跳变端口子集组中的每一个产生至少一个跳变模式，使得所述多个较小的跳变端口子集组中所产生的跳变模式不相交。

8. 如权利要求 1 所述的方法，还包括：

将所述第二数量的跳变端口中的一块分配给用户；以及

为所述一块跳变端口产生所述至少一个跳变模式，使得所述一块跳变端口的跳变模式包括附近的频率子载波。

9. 如权利要求 8 所述的方法，其中所述一块跳变端口的所述至少一个跳变模式包括连续的频率子载波。

10. 如权利要求 1 所述的方法，还包括：

将所述第二数量的跳变端口中的多块分配给用户；以及

为每一块跳变端口产生所述至少一个跳变模式，使得所述跳变模式包括附近的频率子载波。

11. 如权利要求 10 所述的方法，其中所述至少一块跳变端口的所述至少一个跳变模式包括连续的频率子载波。

12. 一种在通信系统中用于产生随机跳变模式的装置，包括：

用于确定第一数量的子载波的模块；

用于确定第二数量的跳变端口的模块；

用于确定第三数量的种子的模块；以及

用于基于所述第一数量的子载波、所述第二数量的跳变端口和基于所述第三数量的种子的伪随机置换产生至少一个跳变模式的模块，

其中用于产生所述至少一个跳变模式的模块包括：

用于利用所述伪随机置换的至少一部分为所述第二数量的跳变端口的至少一个子集组产生第一跳变模式的模块；以及

用于利用所述伪随机置换基于所述第一跳变模式产生第二跳变模式的模块。

13. 如权利要求 12 所述的装置，还包括用于基于系统时间确定所述第三数量的种子中的至少一个的模块。

14. 如权利要求 12 所述的装置，还包括用于基于扇区 ID 确定所述第三数量的种子中的至少一个的模块。

15. 如权利要求 12 所述的装置，还包括用于基于小区 ID 确定所述第三数量的种子中的至少一个的模块。

16. 如权利要求 12 所述的装置，其中所述第一数量大于所述第二数量。

17. 如权利要求 12 所述的装置，还包括用于频繁地改变所述至少一个跳变模式的模块。

18. 如权利要求 12 所述的装置，还包括：

用于将所述第二数量的跳变端口分成多个较小的跳变端口子集组的模块；以及

用于为所述多个较小的跳变端口子集组的每一个产生至少一个跳变模式，使得所述多个较小的跳变端口子集组中所产生的跳变模式不相交的模块。

19. 如权利要求 12 所述的装置，还包括：

用于将所述第二数量的跳变端口中的一块分配给用户的模块；以及

用于为所述一块跳变端口产生所述至少一个跳变模式，使得所述一块跳变端口的跳变模式包括附近的频率子载波的模块。

20. 如权利要求 19 所述的装置，其中所述一块跳变端口的所述至少一个跳变模式包括连续的频率子载波。

21. 如权利要求 12 所述的装置，还包括：

用于将所述第二数量的跳变端口中的多块分配给用户的模块；以及

用于为每一块跳变端口产生所述至少一个跳变模式，使得所述跳变模式包括附近的频率子载波的模块。

22. 如权利要求 21 所述的装置，其中至少一块跳变端口的所述至少一个跳变模式包括连续的频率子载波。

用于在多址通信网络中进行灵活跳变的方法和装置

[0001] 本申请是申请日为2005年12月22日、申请号为200580048263.9(PCT/US2005/046743),发明名称为“用于在多址通信网络中进行灵活跳变的方法和装置”的发明专利申请的分案申请。

[0002] 对相关申请的交叉引用

[0003] 根据35U.S.C.§119(e),本申请要求2004年12月22日递交的,发明名称为“Methods and Apparatus for Flexible Hopping in a Multiple-Access Communication Network”的第60/638,469号美国临时专利申请的优先权,在这里明确地将它全部引入作为参考。

技术领域

[0004] 笼统地说,本发明涉及通信,具体而言,涉及在多址通信网络中产生灵活跳变模式的技术。

背景技术

[0005] 人们广泛部署了通信系统来提供各种通信服务,例如语音、分组数据等等。这些系统可以是时分、频分和/或码分多址系统,通过共享可用系统资源,这些系统能够支持同时与多个用户的通信。这种多址系统的实例包括码分多址(CDMA)系统、多载波CDMA(MC-CDMA)、宽带CDMA(W-CDMA)、高速下行链路分组接入(HSDPA)、时分多址(TDMA)系统、频分多址(FDMA)系统和正交频分多址(OFDMA)系统。

[0006] 通信系统可以采用跳变方案来提高抗干扰能力。因此,在本领域中需要用来有效地设计通信网络中的随机跳变模式的技术。

发明内容

[0007] 公开了用来有效地设计通信系统中的随机跳变模式的技术。所公开的实施例提供了一些方法和系统,用来产生随机跳变模式,频繁地更新这些模式,为不同的小区/扇区产生不同的模式,并且为块跳变产生附近的频率子载波的模式。

附图说明

[0008] 通过下面的详细描述,同时参考附图,本发明的特征和实质将会变得非常清楚,在这些附图中,相似的附图标记表示相同的部件,其中:

[0009] 图1说明一个实施例中的无线接入网;

[0010] 图2画出了一个实施例中的无线接入网框图;

[0011] 图3说明用于产生跳变置换的一个实施例;

[0012] 图4说明Feistel网络;

[0013] 图5说明图4所示Feistel网络中的单独一级;

[0014] 图6说明FLIntraCellHopping是“关闭”的时候,用于产生HijSECTOR(.)的一个

实施例；以及

[0015] 图 7 说明具有端口集合、约束节点和子端口集合的信道树的一个实施例。

具体实施方式

[0016] 在这里用“示例性的”这个词来表示“用作实例或者用于进行说明”。这里描述的任何实施例或者设计都是“示例性的”，不必将它们理解为优选的或者优于其它实施例或设计。

[0017] 图 1 说明具有多个基站 110 和多个终端 120 的无线通信系统 100。基站是与终端通信的站。基站也可以被称为接入点、节点 B 和 / 或一些其它网络实体，并且可以包括它们的一些或全部功能。每个基站 110 都为特定的地理区域 102 提供通信覆盖。“小区”这个术语可以指基站和 / 或它的覆盖区，具体指什么取决于使用这个术语的环境。为了提高系统容量，可以将基站覆盖区划分成多个更小的区域，例如三个更小的区域 104a、104b 和 104c。每个更小的区域由一个相应的基站收发信机子系统 (BTS) 提供服务。“扇区”这个术语可以指 BTS 和 / 或它的覆盖区，具体指什么取决于使用这个术语的环境。对于划分了扇区的小区，这个小区所有扇区的 BTS 通常都一起位于这个小区的基站内。这里描述的发射技术可以被用于具有划分了扇区的小区的系统，也可以被用于具有没有划分扇区的小区的系统。为了简单起见，在以下描述中，“基站”这个术语一般是用于为扇区提供服务的 BTS 以及为小区提供服务的基站。

[0018] 终端 120 通常分布在整个系统内，每个终端都可以是固定的，也可以是移动的。终端也可以被称为移动台、用户设备和 / 或一些其它设备，并且可以具有它们的一些或全部功能。终端可以是无线设备、蜂窝电话、个人数字助理 (PDA)、无线调制解调器卡等等。在任何给定时刻，每个终端都可以在下行链路和上行链路上与零个、一个或多个基站通信。下行链路（或正向链路）是指从基站到终端的通信链路，上行链路（或反向链路）是指从终端到基站的通信链路。

[0019] 对于集中式结构，系统控制器 130 与基站 110 连接，为这些基站提供协调和控制。对于分布式结构，这些基站可以根据需要互相通信。

[0020] 图 2 画出了图 1 所示无线网络 100 中接入点 110x 和接入终端 150x 一个实施例的框图，它们分别实现接入点和接入终端。FL 支持从接入点 110x 到接入终端 150x 的数据传输。RL 支持从接入终端 150x 到接入点 110x 的数据传输。

[0021] 对于正向链路数据传输，接入点 110x、缓冲器 212 从更高层应用接收并保存数据分组。FL TX LP 实体 220 对缓冲器 212 中的数据分组进行处理，提供包含帧的帧序列。MAC/PHY TX 处理器 224 对来自实体 220 的帧序列进行正向链路 MAC 和物理层处理（例如多路复用、编码、调制、加扰、信道化等等），提供数据样本流。发射机单元 (TMTR)226 对来自处理器 224 的数据样本流进行处理（例如转换成模拟信号，进行放大、滤波以及上变频），产生正向链路信号，通过天线 228 发射出去。

[0022] 在接入终端 150x 处，来自接入点 110x 的正向链路信号被天线 262 收到，并且被接收机单元 (RCVR)264 进行处理（例如滤波、放大、下变频和数字化），获得接收样本。MAC/PHY RX 处理器 266 对收到的样本进行正向链路 MAC 和物理层处理（例如去信道化、解扰、解调、解码、去复用等等），提供收到的帧序列。FL RX LP 实体 270 对收到的帧序列进行接收机处

理，提供译码后的数据给重新组装缓冲器 274。FL RX LP 实体 270 还可以为检测到为丢失了的数据产生 NACK，还可以为正确解码的数据产生 ACK。通过反向链路将 NACK 和 ACK 发送给接入点 110x，提供给 FL TX LP 实体 220，后者重新发射丢失了的数据，如果还有的话。重新发射定时器 222 重新发射上一帧来清空缓冲器。NACK 定时器 242 重新发射 NACK。下面将描述这些定时器。

[0023] 对于反向链路数据传输，在接入终端 150x 处，缓冲器 278 从更高层应用接收和保存数据分组。RL TX LP 实体 280 对缓冲器 278 中的数据分组进行处理，提供包含帧的帧序列。MAC/PHY TX 处理器 282 对来自实体 280 的帧序列进行反向链路 MAC 和物理层处理，提供数据样本流。发射机单元 (TMTR)284 处理来自处理器 282 的数据样本流，产生反向链路信号，通过天线 262 将它发射出去。

[0024] 在接入点 110x 处，来自接入终端 150x 的反向链路信号被天线 228 收到，并且被接收机单元 (RCVR)232 处理，获得收到的样本。MAC/PHY RX 处理器 234 对收到的样本进行反向链路 MAC 和物理层处理，提供收到的帧序列。RL RX LP 实体 240 对收到的帧序列进行接收机处理，提供解码后的数据给重新组装缓冲器 242。FL RX LP 实体 240 还可以为被检测为丢失的数据产生 NACK，为正确解码的数据产生 ACK。NACK 和 ACK 通过正向链路发送给接入终端 150x，提供给 RL TX LP 实体 280，后者重新发射丢失的数据，如果还有的话。下面详细描述 FL 和 RL。一般而言，ACK 和 / 或 NACK 反馈可以用链路协议 (LP) 发送，ACK 和 / 或 NACK 反馈也可以用物理层发送。控制器 250 和 290 分别指导在接入点 110x 和接入终端 150x 处的操作。存储器单元 252 和 292 分别保存控制器 250 和 290 使用的程序代码和数据，用于实现所公开的实施例。

[0025] 接入点 110x 可以在正向链路上同时发射数据给一个或多个接入终端。接入终端 150x 可以在反向链路上发射同样的数据给一个或多个接入点。下面的描述针对的是从接入点 110x 到接入终端 150x 的正向链路数据传输，以及从接入终端 150x 到接入点 110x 的反向链路数据传输。

[0026] 可以用跳变置换来将跳变端口集合映射到子载波集合。在一个实施例中，可以从 NFFT-NGUARD 到 NFFT-1 给其编制下标的跳变端口可以用跳变置换映射到保护载波集合。如果这些载波没有被调制，就可以不指定这个映射的各个元素。可以将这个跳变序列描述为从按照 0 到 NFFT-NGUARD-1 编号的跳变端口集合映射到可用子载波的集合，例如除了这个保护子载波集合以外的所有子载波的集合。

[0027] 令 Hij(p) 是与超帧下标“i”中第 j 个调制码元的跳变端口下标“p”对应的子载波下标。在这里，p 是 0 和 NFFT-NGUARD-1 之间的一个下标，j 是大于 4 的整数。没有为超帧前序中的码元定义任何跳变置换。Hij(p) 是 NGUARD/2 和 NFFT-NGUARD/2-1 之间的一个值，可以按照以下程序来计算它：

[0028] $Hij(p) = NGUARD/2 + HijGLOBAL(HijSECTOR(p))$

[0029] 其中 HijGLOBAL(.) 和 HijSECTOR(.) 是集合 $\{0,1,2,\cdots,NFFT-NGUARD-1\}$ 的置换。

[0030] HijGLOBAL(.) 是可以不依赖于 SECTOR_PN_OFFSET 的置换，而 HijSECTOR(.) 则是可以依赖于 SECTOR_PN_OFFSET 的置换。对于 FLSectorHopSeed 具有相同值的两个扇区，HijGLOBAL 可以相同。对于不同的扇区，HijSECTOR 可以不同，除非将变量

FLIntraCellCommonHopping置位。此外,HijSECTOR(.)将端口集合内的跳变端口映射到这个端口集合内的跳变端口。从信道树确定端口集合的数量及其大小,这可以从FTC MAC协议来确定。

[0031] 假设有编号为0,1,…,K-1的K个端口集合。假设第k个端口集合中跳变端口的数量为Nk,保护区域中的跳变端口被排除在外。如果只有一个端口集合,用0编号,那么N0 = NFFT-NGUARD。依赖于扇区的置换HijSECTOR(.)可以将第0个端口集合中的跳变端口,也就是编号为{0,1,2,…,N0-1}的跳变端口,映射到同一集合中的编号。将这个映射表示为P0ij(.)。因此,如果p在第0个跳变端口集合中,那么HijSECTOR(p) = P0ij(p)。类似地,依赖于扇区的置换可以将第一个端口集合中的跳变端口,也就是编号为{N0,N0+1,N0+2,…,N0+N1-1}的跳变端口,映射到同一集合中的编号。这是通过利用表示为P1ij(.)的对{0,1,2,…,N1-1}的置换完成的。因此,如果p在第一个端口集合中,那么HijSECTOR(p) = N0+P1ij(p-N0)。类似地,如果p在第二个端口集合中,那么HijSECTOR(p) = N0+N1+P2ij(p-N0-N1)。因此,HijSECTOR(.)由总共K个端口内集合置换P0ij(.),P1ij(.),…,PK-1ij(.)定义。

[0032] 根据一个实施例,跳变序列生成中的一个元素是Feistel网络。三级Feistel网络产生尺寸是2的幂的伪随机置换。产生{0,1,2,…,2n-2,2n-1}的置换π(x)的Feistel网络按照如下方式工作:

[0033] 1.将n比特的输入x分裂成两个部分(L,R),每个部分包括大致相同数量的比特。如果n是偶数,那么L可以是x的最高n/2位,R可以是最低n/2位。如果n是奇数,那么L可以是x的最高(n-1)/2位,R可以是x的最低(n+1)/2位。

[0034] 2.Feistel网络第一级的输出π1(x)是一个(R,L□f(R))形式的n比特量。在这里,f(R) = (R+S1)mod 2|L|,其中|L|是L中的比特数,S1是|L|比特的种子,□是按位XOR操作。可以基于系统时间、sector_ID、Cell_ID和/或扇区PN_offset来产生种子。

[0035] 3.将输出π1(x)输入Feistel网络的下一级,这个下一级可以与第一级相同,只有使用的种子是S2除外。将第二级的输出π2(π1(x))输入第三级,这个第三级可以与前两级相同,只有使用的种子是S3除外。第三级π3(π2(π1(x)))的输出是最后的输出π(x)。

[0036] 图4说明一个三级Feistel网络。图5说明n = 9的情况下的单独一个Feistel级。根据一个实施例,要在超帧中第j个码元处使用的全局置换Hijglobal(.)可以按照如下方式从初始置换HiGLOBAL(.)产生:

[0037] 1.HijGLOBAL(x) = HiGLOBAL(j+HiGLOBAL(j+x)),其中两个加法运算都可以按照模(NFFT-NGUARD)进行。可以按照以下程序产生初始置换HiGLOBAL(.):

[0038] 2.找出使得NFFT ≤ 2n的最小整数n。如果n是偶数,则设置|L| = n/2;如果n是奇数,则设置|L| = (n-1)/2。

[0039] 3.按照如下方式设置Feistel种子S1、S2和S3:

[0040] 4.找出S' = [(FLSectorHopSeed*4096+(i mod 4096))*2654435761]mod 232。将S设置为S'的32位表示中各个比特相反的值。

[0041] 5.将S1设置为S的最低|L|位,将S2设置为S接下来的|L|个低位,将S3设置为S再接下来的|L|个低位。换句话说,S1 = Smod 2|L|,S2 = (S-S1)/2|L|mod 2|L|,S3

= (S-S1-S22|L|)/22|L|mod 2|L|。

[0042] 6. 将两个计数器 x 和 y 初始化成 0。

[0043] 7. 找出以 S1、S2 和 S3 为种子的 Feistel 网络的输出 π(x)。

[0044] 如果 π(x) 小于 (NFFT - NGUARD),设置 Higlobal(y) = π(x),将 y 加 1。将计数器 x 加 1。如果 x<NFFT,重复,否则停止。

[0045] 可以针对“FLIntraCellCommonHopping”的不同值分别描述 HijSECTOR 的产生。如果 FLIntraCellCommonHopping 是“关闭”,那么可以从初始置换 Pki(.) 按照程序 Pkij(x) = Pki(αj+Pki(βj+x)) 产生构成 HijSECTOR(.) 的 K 个端口集合内置换 Pkij(.),其中两个加法运算都按照模 Nk 进行。αj 和 βj 都是利用具有发生器多项式 h(D) = D18+D11+1 的 PN 寄存器产生的 9 比特随机数。按照以下方式产生 αj 和 βj 这两个数:

[0046] 1. 将 SECTOR_PN_OFFSET 与超帧下标 i 的最低 12 位进行 XOR 运算,获得表示为 Boff 的 12 比特数 [b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0]。

[0047] 2. 在超帧的开头将 PN 寄存器初始化成 [111111 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0]。

[0048] 3. 然后每个码元给寄存器 18 个时钟信号。在码元 j 之前寄存器的内容决定了 αj 和 βj,其中 αj 被设置成寄存器的最高 9 位,βj 被设置成寄存器的最低 9 位。(因此,α0 = [111111 b11 b10 b9], β0 = [b8 b7 b6 b5 b4 b3 b2 b1 b0])。

[0049] 参考图 6,按照以下程序产生初始置换 Pki(.):

[0050] (1) 找出使得 NFFT ≤ 2n 的最小整数 n。如果 n 是偶数,则设置 |L| = n/2 ;如果 n 是奇数,则设置 |L| = (n-1)/2。

[0051] (2) 按照如下方式设置 Feistel 种子 S1、S2 和 S3:

[0052] (3) 找出 S' = [Boff.*2654435761]mod 232。将 S 设置为 S'的 32 位表示中各个比特相反的值。

[0053] (4) 将 S1 设置为 S 的最低 |L| 位,将 S2 设置为 S 接下来的 |L| 个低位,将 S3 设置为 S 再接下来的 |L| 个低位。换句话说,S1 = Smod 2|L|,S2 = (S-S1)/2|L|mod 2|L|,S3 = (S-S1-S22|L|)/22|L|mod 2|L|。

[0054] (5) 将 K 个计数器 y0、y1、……、yK-1 初始化成 0。将另一个计数器 x 初始化成 0。

[0055] (6) 找出以 S1、S2 和 S3 为种子的 Feistel 网络的输出 π(x)。

[0056] (7) 如果 π(x) 对应于第 k 个端口集合中的一个跳变端口(也就是如果 N0+N1+……+Nk-1 ≤ π(x)<N0+N1+……Nk-1+Nk),那么:

[0057] (8) 设置 Pki(yk) = π(x)-(N0+N1+……+Nk-1),并且

[0058] (9) 将 yk 加 1。

[0059] (10) 将计数器 x 加 1。如果 x<NFFT,重复步骤 4 ～ 6,否则停止。

[0060] 当 FLIntraCellCommonHopping 是“打开”的时候,可以从初始置换 Pki(.) 按照程序 Pkij(x) = Pki(αj+Pki(βj+x)) 产生构成 HijSECTOR(.) 的 K 个端口集合内置换 Pkij(.),其中两个加法运算都按照模 Nk 进行。αj 和 βj 都是利用具有发生器多项式 h(D) = D18+D11+1 的 PN 寄存器产生的 9 比特随机数。按照以下方式产生 αj 和 βj 这两个数:

[0061] 1. 将 SECTOR_PN_OFFSET 与超帧下标 i 的最低 12 位进行逐位 XOR 运算,获得表示

为Boff的12比特数[b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0]。

[0062] 2.在超帧的开头将PN寄存器初始化成[111111 b11 b10 b9 b8 i7 i6 i5 b4 b3 b2 b1 b0],其中i7i6i5是超帧下标i的第7、6和5位。将12比特数[b11 b10 b9 b8 i7 i6 i5 b4 b3 b2 b1 b0]表示为Bon。

[0063] 3.然后每个OFDM码元给寄存器18个时钟信号。在OFDM码元j之前寄存器的内容决定了αj和βj,其中αj被设置成寄存器的最高9位,βj被设置成寄存器的最低9位。(因此,α0=[111111 b11 b10 b9],β0=[b8 i7 i6 i5 b4 b3 b2 b1 b0])。

[0064] 当FLIntraCellCommonHopping是“打开”的时候,基于Bon产生除了具有下标0的端口集合以外的所有端口集合的初始置换Pki(.),而端口集合下标0的初始置换则基于Boff产生。为了正确地使用这一模式,同一小区的两个扇区的SECTOR_PN_OFFSET可以有三个比特位置不同,也就是说具有下标5、6和7的比特位置不同。在这里,位下标0对应于最高位,而位下标11对应于最低位。

[0065] 为除了具有下标0的那个以外的所有端口集合产生初始置换的程序如下:

[0066] 1.找出使得NFFT≤2n的最小整数n。如果n是偶数,则设置|L|=n/2;如果n是奇数,则设置|L|=(n-1)/2。

[0067] 2.按照如下方式设置Feistel种子S1、S2和S3:

[0068] 3.找出S'=[Bon*2654435761]mod 232。将S设置为S'的32位表示中各个比特相反的值。

[0069] 4.将S1设置为S的最低|L|位,将S2设置为S接下来的|L|个低位,将S3设置为S再接下来的|L|个低位。换句话说,S1=S mod 2|L|,S2=(S-S1)/2|L|mod 2|L|,S3=(S-S1-S22|L|)/22|L|mod 2|L|。

[0070] 5.将K个计数器y0、y1、……、yK-1初始化成0。将另一个计数器x初始化成0。

[0071] 6.找出以S1、S2和S3为种子的Feistel网络的输出π(x)。

[0072] 7.如果对于k>0,π(x)对应于第k个端口集合中的一个跳变端口(也就是如果N0+N1+……+Nk-1≤π(x)<N0+N1+……Nk-1+Nk),那么:

[0073] 8.设置Pki(yk)=π(x)-(N0+N1+……+Nk-1),并且

[0074] 9.将yk加1。

[0075] 10.将计数器x加1。如果x<NFFT,重复步骤4～6,否则停止。

[0076] 11.按照如下方式产生端口集合下标0的初始置换:

[0077] 12.找出使得(NFFT-NGUARD)≤2n的最小整数n。如果n是偶数,则设置|L|=n/2;如果n是奇数,则设置|L|=(n-1)/2。

[0078] 13.按照如下方式设置Feistel种子S1、S2和S3:

[0079] 14.找出S'=[Boff*2654435761]mod 232。将S设置为S'的32位表示中各个比特相反的值。

[0080] 15.将S1设置为S的最低|L|位,将S2设置为S接下来的|L|个低位,将S3设置为S再接下来的|L|个低位。换句话说,S1=S mod 2|L|,S2=(S-S1)/2|L|mod 2|L|,S3=(S-S1-S22|L|)/22|L|mod 2|L|。

[0081] 16.将两个计数器x和y初始化成0。

[0082] 17.找出以S1、S2和S3为种子的Feistel网络的输出π(x)。

[0083] 18. 如果 π(x) 对应于第 0 个端口集合中的一个跳变端口（也就是如果 π(x)<N0)，那么：

[0084] 19. 设置 P0i(y) = π(x)，并且

[0085] 20. 将 y 加 1。

[0086] 21. 将计数器 x 加 1。如果 x<NFFT，重复步骤 4 ～ 6，否则停止。

[0087] 共用导引信道 (F-CPICH) 可以占据每 PHY 帧的每个调制码元中均匀间隔的一个子载波集合。令 Np 是每个 OFDM 码元中导引子载波的标称数量。Np 由“SystemInfo”块的“导引信号数量”字段给出，这个块是系统开销消息协议的公共数据。然后，相邻导引子载波之间的间隔可以等于 Dp = NFFT/Np。

[0088] 对于 PHY 帧中的每个码元，可以利用以下程序来确定在 0 和 Dp-1 之间取值的变量 Offsetp：令 i 是超帧下标，令 j 是这个超帧内 OFDM 码元的下标（以下标 0 开始）。如果 j ≤ 4，也就是如果码元在超帧前序（中，就不定义变量 Offsetp。

[0089] 如果 j 是奇数，就可以利用具有发生器多项式 h(D) = D13+D12+D11+D8+1 的 13 位 PN 寄存器确定 Offsetp。可以在超帧开头将移位寄存器初始化到状态 [1 p11 p10 p9 p8 p7 p6 p5 p4 p3 p2 p1 p0]，其中 p11、p10、p9、……、p0 是 SECTOR_PN_PHASE 的 12 位，p11 是最高位，p0 是最低位。每个码元可以给移位寄存器 13 个时钟脉冲。可以将 Offsetp 选择为寄存器模 Dp 的值。在这里，寄存器的值是码元 j 之前的值，也就是经过了 j*13 次时钟脉冲以后寄存器的值。

[0090] 如果 j 是偶数，就可以通过将 Dp/2 这个值加到前一个 OFDM 码元模 Dp 的 Offsetp 的值上去来计算 Offsetp 的值。对于 PHY 帧中的每个码元，如果满足如下两个条件，下标是 isc 的子载波就可以由 F-CPICH 占据：isc mod NFFT = Offsetp，并且具有下标 isc 的子载波不是保护子载波。

[0091] 可以用复值 $(\sqrt{P}, 0)$ 来调制 F-CPICH 占据的每个子载波，其中 P 是 F-CPICH 的功率谱密度与 F-ACQCH 中第二码元的功率谱密度之比。这个比值由 SystemInfo 块的“CommonPilotPower”字段给出，它可以是系统开销消息协议的公共数据。

[0092] 根据一个实施例，反向链路可以实现块跳变，也就是将跳变端口集合划分成 NBLOCK 个跳变端口的块，它可以是连续方式的。跳变端口 0、1、……、NBLOCK-1 形成块 0，跳变端口 NBLOCK、NBLOCK+1、……、2NBLOCK － 1 形成块 1，等等。一块中连续的跳变端口被跳变模式映射到连续的子载波，也就是说，如果将跳变端口 0 映射到子载波 i，那么跳变端口 1 被映射到子载波 i+1，跳变端口 2 被映射到子载波 i+2，等等。对于长数据段，NBLOCK 的值可以是 8，对于短数据段，NBLOCK 的值可以是 TBD。可以针对长和短数据段分别描述跳变序列。

[0093] 保护载波的数量 NGUARD 可以是 NBLOCK 的整数倍。如上所述，可以通过跳变置换将下标从 NFFT-NGUARD 到 NFFT-1 的跳变端口映射到上述保护载波集合。由于这些载波不是调制载波，因此没有指定这个映射的各个元素。可以将跳变序列描述为从跳变端口号从 0 到 NFFT-NGUARD-1 的集合到可用子载波集合（也就是除了保护子载波集合以外的全部）的映射。

[0094] 跳变序列生成中的基本元素可以是 Feistel 网络。三级 Feistel 网络产生大小为

2 的幂的伪随机置换。产生 {0,1,2,…,2n-2,2n-1} 的置换 π(x) 的 Feistel 网络按照如下方式工作：

[0095] 1. 将 n 比特的输入 x 分裂成两个部分 (L,R)，每个部分包括大致相同数量的比特。如果 n 是偶数，那么 L 可以是 x 的最高 n/2 位，R 可以是最低 n/2 位。如果 n 是奇数，那么 L 可以是 x 的最高 (n-1)/2 位，R 可以是 x 的最低 (n+1)/2 位。

[0096] 2. Feistel 网络第一级的输出 π1(x) 是一个 (R,L□f(R)) 形式的 n 比特量。在这里，f(R) = (R+S1)mod 2|L|，其中 |L| 是 L 中的比特数，S1 是 |L| 比特的种子，□是按位 XOR 操作。

[0097] 3. 将输出 π1(x) 输入 Feistel 网络的下一级，这个下一级与第一级相同，只有使用的种子是 S2 除外。将第二级的输出 π2(π1(x)) 输入第三级，这个第三级与前两级相同，只有使用的种子是 S3 除外。第三级 π3(π2(π1(x))) 的输出是最后的输出 π(x)。

[0098] 图 4 说明一个三级 Feistel 网络，图 5 说明 n = 9 的情况下的单独一个 Feistel 级。长数据段支持受约束的跳变。信道树可以将节点集合定义为约束节点，跳变序列确保是约束节点的一部分的所有跳变端口的集合被映射到连续的子载波集合。可以将连续的跳变端口映射到连续子载波，也可以不这样。

[0099] 为了支持受约束的跳变，对信道树施加如下限制：

[0100] (1) 约束节点可以满足以下要求：

[0101] a. 至少有两个约束节点。

[0102] b. 包括约束节点和它们的祖先的子图可以是二叉树。

[0103] (2) 任何基节点都可以有也只能有一个约束节点作为祖先。

[0104] (3) 端口集合中的所有节点都可以有共同的祖先，所述端口集合可以是这个祖先的所有子孙的集合。

[0105] 具有比约束节点多的作为子孙的端口集合可以被分裂成子端口集合，每个约束节点都定义所述子端口集合。可以按照升序将子端口集合编号为 {0,1,…,K-1}，也就是说子端口集合 0 可以包含编号最小的跳变端口，子端口集合 K-1 可以包含编号最大的跳变端口。

[0106] 参考图 7，描述具有端口集合、约束节点和子端口集合的信道树。令 Hij'(p') 表示分配给超帧 I 中编号为 j' 的调制码元里跳变端口 p' 的频率，其中 j' 被约束成位于长数据段中。在这里 p' 是 0 和 NFFT-NGUARD-1 之间的一个下标，Hij'(p') 是 NGUARD/2 和 NFFT-NGUARD/2-1 之间的一个值，可以按照以下公式来计算它：Hij'(p') = NGUARD/2+NBLOCK*(HijGLOBAL(k)+HijkSECTOR(p))+(p' mod NBLOCK)。

[0107] 在这里，$p=\lfloor p'/N_{BLOCK} \rfloor$ 表示包含跳变端口 p' 的跳变端口块，k 表示包含跳变端口 p' 的子端口集合，j 表示与码元 j' 对应的超帧内的跳变间隔下标。跳变间隔下标在超帧内顺序计数，但忽略控制段，也就是跳变间隔 0 和 1 属于超帧里的第一帧，跳变间隔 2 和 3 属于超帧里的第二帧，等等。HijSECTOR(.) 是依赖于扇区的函数，它置换第 k 个子端口集合中的跳变端口块。HijGLOBAL(k) 是置换频率周围的子端口集合的函数（或者是一个扇区一个扇区地，或者是独立于扇区地）。

[0108] 可以单独为 RLIntraCellCommonHopping 的不同值描述 HijSECTOR 的生成。首先，存在 RLIntraCellCommonHopping 是“关闭”的时候。在这种情况下，令 K 是子端口集合的

总数,Nk 是第 k 个子端口集合中跳变端口块的数量(除了保护区域中的跳变端口块以外)。跳变端口块的数量是跳变端口的数量除以 NBLOCK。将感兴趣的扇区的 SECTOR_PN_OFFSET 与超帧下标 i 的最低 12 位逐位 XOR,获得表示为 Boff 的 12 位数 [b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0]。可以利用它按照以下程序来生成置换 HijSECTOR(.):

[0109] (1) 找出使得 NFFT ≤ 2n 的最小整数 n。如果 n 是偶数,则设置 |L| = n/2 ;如果 n 是奇数,则设置 |L| = (n-1)/2。

[0110] (2) 按照如下方式设置 Feistel 种子 S1、S2 和 S3 :

[0111] (30 找出 S' = [(Boff*32+j)*2654435761]mod 232。将 S 设置为 S'的 32 位表示中各个比特相反的值。将 S1 设置为 S 的最低 |L| 位,将 S2 设置为 S 接下来的 |L| 个低位,将 S3 设置为 S 再接下来的 |L| 个低位。换句话说,S1 = S mod 2|L|,S2 = (S-S1)/2|L|mod 2|L|,S3 = (S-S1-S22|L|)/22|L|mod 2|L|。

[0112] 将计数器 x 初始化成 0。将 K 个计数器 y0,y1,y2,…,yK-1 分别初始化成 0,N0,N0+N1,N0+N1+N2,...,N0+N1+...+NK-2。(这些初值对应于所述子端口集合中编号最小的跳变端口块)

[0113] 找出以 S1、S2 和 S3 为种子的 Feistel 网络的输出 π(x)。

[0114] 如果 π(x) 对应于第 k 个子端口集合中的一个跳变端口块,也就是如果 N0+N1+……+Nk-1 ≤ π(x)<N0+N1+……+Nk,那么 :

[0115] 设置 HijSECTOR(yk) = π(x),并且

[0116] 将 yk 加 1。

[0117] 将计数器 x 加 1。如果 x<NFFT,重复步骤 4 ～ 6,否则停止。

[0118] RLIntraCellCommonHopping 是“打开的”

[0119] 令 K 是子端口集合的总数,Nk 是第 k 个子端口集合中跳变端口块的数量,除了保护区域中的跳变端口块以外。跳变端口块的数量是跳变端口数量除以 NBLOCK。

[0120] 将所述扇区的 PN 偏移与超帧下标 i 的最低 12 位逐位 XOR,获得表示为 Boff 的 12 位数 [b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0]。将 12 位数 [b11 b10 b9 b8 i7 i6 i5 b4 b3 b2 b1 b0] 表示为 Bon,其中 i7 i6 i5 是超帧下标 i 的第 7、6 和 5 位。

[0121] 当 RLIntraCellCommonHopping 是“打开的”的时候,可以用 Bon 来产生不是端口集合 0 一部分的子端口集合的置换 HijkSECTOR(.),而 Boff 则可以被用来产生是子端口集合 0 一部分的子端口集合的置换 HijkSECTOR(.)。同一小区的两个扇区的 SECTOR_PN_OFFSET 可能有三位的位置不同,也就是下标为 5、6 和 7 的那些位。在这里,位下标 0 对应于最高位,而位下标 11 则对应于最低位。

[0122] 对于不是端口集合 0 一部分的子端口集合,可以按照如下程序来生成 HijkSECTOR(.):

[0123] 找出使得 NFFT ≤ 2n 的最小整数 n。如果 n 是偶数,则设置 |L| = n/2 ;如果 n 是奇数,则设置 |L| = (n-1)/2。

[0124] 按照如下方式设置 Feistel 种子 S1、S2 和 S3 :

[0125] 找出 S' = [(Bon*32+j)*2654435761]mod 232。将 S 设置为 S' 的 32 位表示中各个比特相反的值。

[0126] 将 S1 设置为 S 的最低 |L| 位,将 S2 设置为 S 接下来的 |L| 个低位,将 S3 设置为 S

再接下来的 |L| 个低位。换句话说，S1 = S mod 2|L|，S2 = (S-S1)/2|L|mod 2|L|，S3 = (S-S1-S22|L|)/22|L|mod 2|L|。

[0127] 将计数器 x 初始化成 0。将 K 个计数器 y0,y1,y2,…,yK-1 分别初始化成 0,N0,N0+N1,N0+N1+N2,...,N0+N1+...+NK-2。这些初值对应于所述子端口集合中编号最小的跳变端口块。

[0128] 找出以 S1、S2 和 S3 为种子的 Feistel 网络的输出 π(x)。

[0129] 如果 π(x) 对应于第 k 个子端口集合中的一个跳变端口块，(也就是如果 N0+N1+……+Nk-1 ≤ π(x)<N0+N1+……+Nk,) 并且第 k 个子端口集合是端口集合 0 的一部分，那么：

[0130] 设置 HijSECTOR(yk) = π(x)，并且

[0131] 将 yk 加 1。

[0132] 将计数器 x 加 1。如果 x<NFFT，重复步骤 4 ～ 6，否则停止。

[0133] 对于是端口集合 0 一部分的子端口集合，可以按照如下程序来生成 HijkSECTOR(.)：

[0134] 找出使得 NFFT ≤ 2n 的最小整数 n。如果 n 是偶数，则设置 |L| = n/2；如果 n 是奇数，则设置 |L| = (n-1)/2。

[0135] 按照如下方式设置 Feistel 种子 S1、S2 和 S3：

[0136] 找出 S' = [(Boff*32+j)*2654435761]mod 232。将 S 设置为 S'的 32 位表示中各个比特相反的值。

[0137] 将 S1 设置为 S 的最低 |L| 位，将 S2 设置为 S 接下来的 |L| 个低位，将 S3 设置为 S 再接下来的 |L| 个低位。换句话说，S1 = S mod 2|L|，S2 = (S-S1)/2|L|mod 2|L|，S3 = (S-S1-S22|L|)/22|L| mod 2|L|。

[0138] 将计数器 x 初始化成 0。将 K 个计数器 y0,y1,y2,…,yK-1 分别初始化成 0,N0,N0+N1,N0+N1+N2,...,N0+N1+...+NK-2。(这些初值对应于所述子端口集合中编号最小的跳变端口块)

[0139] 找出以 S1、S2 和 S3 为种子的 Feistel 网络的输出 π(x)。

[0140] 如果 π(x) 对应于第 k 个子端口集合中的一个跳变端口块，(也就是如果 N0+N1+……+Nk-1 ≤ π(x)<N0+N1+……+Nk,) 并且第 k 个子端口集合是 111 端口集合的一部分，那么：

[0141] 设置 HijkSECTOR(yk) = π(x)，并且

[0142] 将 yk 加 1。

[0143] 将计数器 x 加 1。如果 x<NFFT，重复步骤 4 ～ 6，否则停止。

[0144] HijGLOBAL(.) 的生成

[0145] HijGLOBAL(k) 可以置换 K 个子端口集合，提高频率分集，而没有多少或者根本没有干扰分集中的损失。可以按照以下程序来做到这一点：

[0146] 按照以下规则来产生种子 S：

[0147] 如果有一个以上的端口集合，那么 S' = [(RLSectorHopSeed*4096*32+(i mod 4096)*32+j)*2654435761]mod 232

[0148] 如果只有一个端口集合，那么 S' = [(Boff*32+j)*2654435761]mod 232

[0149] S 是 S' 的 32 位表示中各个比特相反的值。

[0150] 可以将两个深度 1 节点（也就是根节点的子节点）标为 A 和 B，KA 可以是作为 A 的子孙的子端口集合的数量，KB 可以是作为 B 的子孙的子端口集合的数量。(KA+KB = K)。

[0151] 对 {0,1,...,KA-1} 的置换可以按照字母顺序列出来，编号为 0 到 (KA ! -1)，其中对于任意正整数，k ! 表示乘积 k(k-1)(k-2)...2。例如，如果 KA = 3，那么顺序是 012,021,102,120,201,210，编号从 1 到 5。可以将编号为 $\lfloor S/2 \rfloor \bmod K_A!$ 的置换选为作为 A 的子孙的子端口集合的置换 PA。

[0152] 类似地，对 {KA,KA+1,...,KA+KB-1} 的置换可以按照字母顺序列出来，编号为 0 到 (KB ! -1)。例如，如果 KA = 3 并且 KA = 2，那么置换是 34 和 43，分别编号为 0 和 1。可以将编号为 $\lfloor S/2 \rfloor \bmod K_B!$ 的置换选为作为 B 的子孙的子端口集合的置换 PB。

[0153] 按照如下方式确定集合 {A,B} 的置换 ：

[0154] 如果 j 是偶数，那么，如果 S mod 2 = 0，置换可以是 AB，如果 S mod 2 = 1，置换可以是 BA。

[0155] 如果 j 是奇数，置换可以是在跳变间隔 j-1 处选择的置换的相反。

[0156] 因此，子端口集合的整个置换可以是 PAPB 或者 PBPA。例如，如果 PA = 021 并且 PB = 43，而且选择了 AB，那么整个置换可以是 02143。如果选择了 BA，它就会是 43021。

[0157] 一旦最后定下子端口集合的置换，就可以通过从置换以后同一跳变端口块的位置减去置换之前子端口集合中的最小编号的跳变端口块的位置，来计算函数 HijGLOBAL(k)。例如，如果子端口集合置换是 02143，那么 ：

[0158] HijGLOBAL(0) = (0)-(0)。

[0159] HijGLOBAL(1) = (N0+N2)-(N0)。

[0160] HijGLOBAL(2) = (N0)-(N0+N1)。

[0161] HijGLOBAL(3) = (N0+N2+N1+N4)-(N0+N1+N2)。

[0162] HijGLOBAL(4) = (N0+N2+N1)-(N0+N1+N2+N3)。

[0163] 其中 Nk 是第 k 个子端口集合中跳变端口块的数量。

[0164] 在一个实施例中，用于产生随机跳变模式的一种系统和方法包括确定第一数量的子载波和第二数量的跳变端口。因为保护频带的存在，这些保护频带要占用一些子载波，因此，跳变端口的数量可能少于子载波的数量。如上所述，这一过程还包括确定第三数量的种子。如上所述，这一过程基于第一数量的子载波、第二数量的跳变端口和第三数量的种子，例如利用 Feistel 网络，产生至少一个跳变模式。这些种子可以基于系统时间、扇区 ID、小区 ID 或者它们的组合来加以确定。

[0165] 在一个实施例中，可以频繁地更新或改变所产生的跳变模式，以确保频率分集。更新可以基于系统时间。更新还可以包括每个预定量的时间改变跳变端口实体的子载波频率的预定量。

[0166] 在一个实施例中，可以将跳变端口组成较小的跳变端口组，将每一组输入 Feistel 网络的一部分 / 一个单元，从而为每一个较小组跳变端口产生至少一个跳变模式。在这种情况下，每一组子载波都可以对应于相同或不同小区中的不同扇区，降低干扰。

[0167] 在一个实施例中，可以将一块（例如连续的）跳变端口分配给用户。例如，为了方

便信道估计，为所述块跳变端口产生的跳变模式可以包括附近的频率子载波和 / 或连续的频率子载波。

[0168] 在一个实施例中，可以将多块跳变端口分配给用户。可以将所述块跳变端口的对应跳变模式放置在所希望的附近。例如，为了确保频率分集和较低干扰，可以让所述多块跳变端口的跳变模式互相分开。但是，如果所述多块跳变端口的跳变模式互相相隔太远，带外频谱发射会增大。

[0169] 在一个实施例中，为多个跳变端口产生随机跳变模式的一种方法包括在树的第一层（树叶）按顺序给跳变端口实体（跳变端口和 / 或多块跳变端口）排序，如果至少满足第一条件，就在更低层交换每一对跳变端口实体，从而产生更高层跳变端口实体。重复这一操作，如果至少满足第二条件，就在更高层交换每一对跳变端口实体。重复这一过程，直到到达树顶，产生随机跳变模式。这些跳变端口实体可以包括至少一块连续的跳变端口，它们可以对应于连续块子载波频率。

[0170] 例如，考虑编号为 0、1、2 和 3 的一个跳变端口实体集合。在最低层，存在跳变端口对 0-1 和 2-3。如果有一对满足第一条件，例如抛掷硬币等待正面出现，就交换这一对。例如，0-1 这一对可以不交换，但是 2-3 这一对可以交换，得到更高层跳变端口实体 0-1 和 3-2。现在，重复这一过程，如果满足第二或者同样的条件，就交换更高层对 (0-1 和 3-2)。例如，这个更高层对可以交换，得到跳变模式 3、2、0 和 1。要注意，这个过程中可以包括具有任意数量的跳变端口的任意数量的跳变端口实体。

[0171] 可以将公开的上述实施例应用于以下技术中的任意技术或者技术组合 :码分多址 (CDMA) 系统、多载波 CDMA(MC-CDMA)、宽带多址 (TDMA) 系统、频分多址 (FDMA) 系统和正交频分多址 (OFDMA) 系统。

[0172] 在这里描述的信令发射技术可以用各种方式来实现。例如，这些技术可以用硬件、软件或者它们的组合来实现。对于硬件实现，用于处理（例如压缩和编码）信令的处理单元可以在一个或多个专用集成电路 (ASIC)、数字信号处理器 (DSP)、数字信号处理装置 (DSPD)、可编程逻辑器件 (PLD)、现场可编程门阵列 (FPAG)、处理器、控制器、微控制器、微处理器、设计成实现这里描述的功能的其它电子单元或者它们的组合中实现。用于对信令进行译码和解压缩的处理单元也可以用一个或多个 ASIC、DSP 等等来实现。

[0173] 对于软件实现，信令发射技术可以用实现这里描述的功能的模块（例如子程序、函数等等）来实现。可以将软件代码储存在存储器单元（例如图 2 中的存储器单元 252 或 292）中，由处理器（例如控制器 250 或 290）来执行。存储器单元可以在处理器内或者在处理器外实现。

[0174] 给出前面对公开的实施例的描述是为了让本领域技术人员能够制造或使用本发明。对这些实施例的各种变形对于本领域技术人员而言将是显而易见的，这里给出的一般原理可以被应用于其它实施例而不会偏离本发明的实质和范围。因此，本发明不是局限于这里公开的实施例，而是与这里公开的原理和新颖特征的最大范围一致。

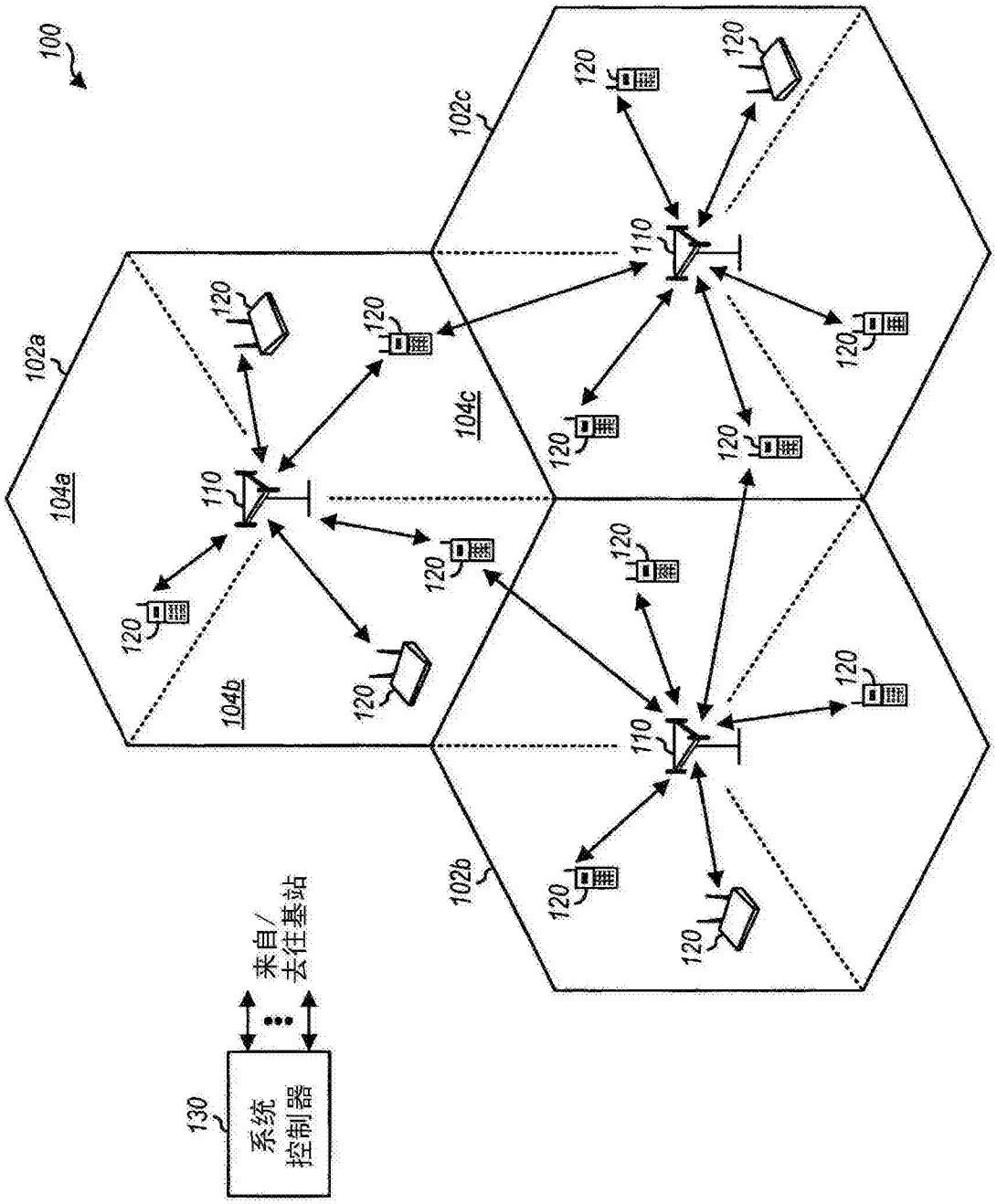


图 1

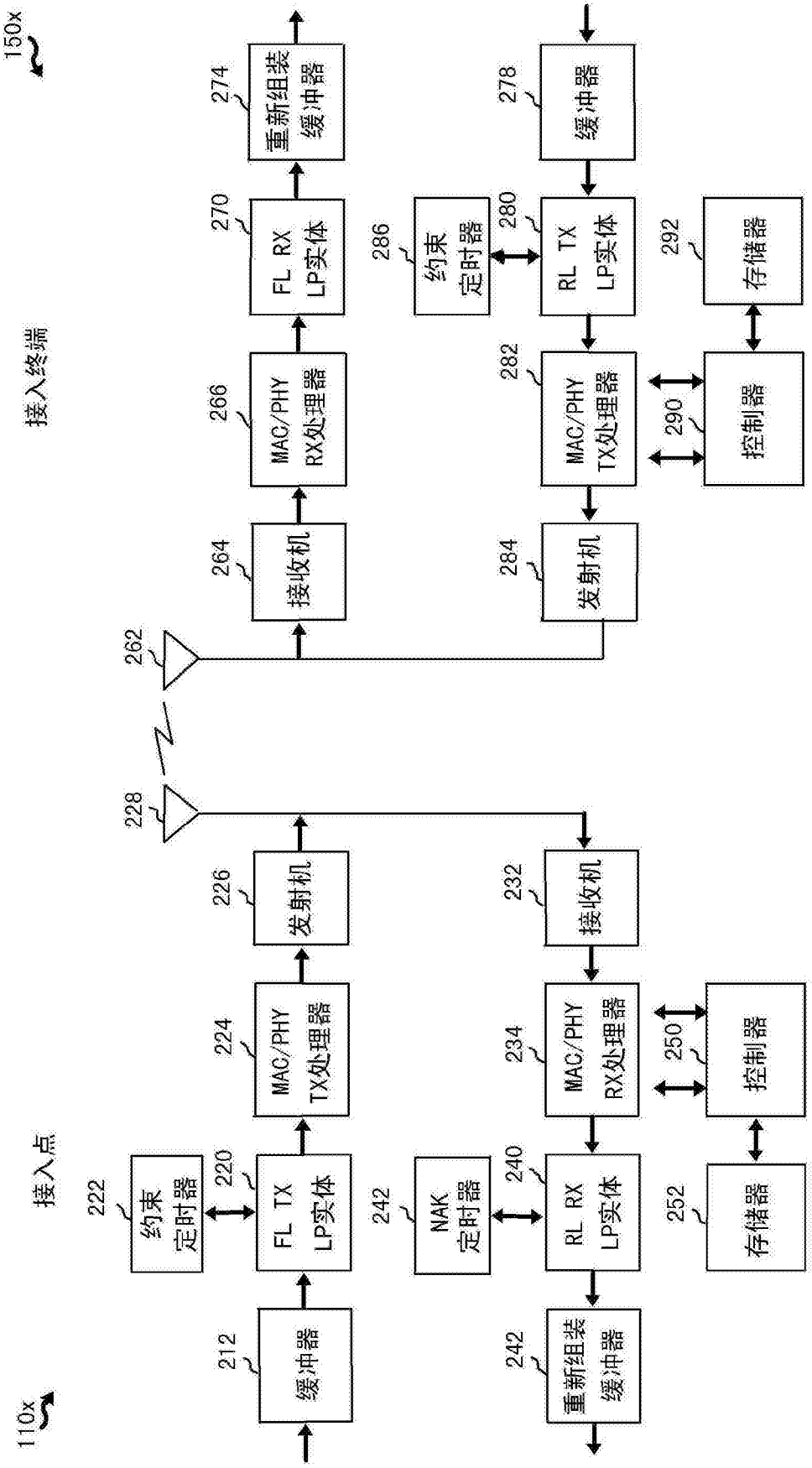

150x
接入终端
274
重新组装缓冲器
270
FL RX LP实体
266
MAC/PHY RX处理器
264
接收机
262
228
226
发射机
224
MAC/PHY TX处理器
222
约束定时器
220
FL TX LP实体
212
缓冲器
110x
接入点
278
缓冲器
286
约束定时器
280
RL TX LP实体
292
存储器
282
MAC/PHY TX处理器
290
控制器
284
发射机
232
接收机
234
MAC/PHY RX处理器
250
控制器
242
NAK定时器
240
RL RX LP实体
252
存储器
242
重新组装缓冲器

图 2

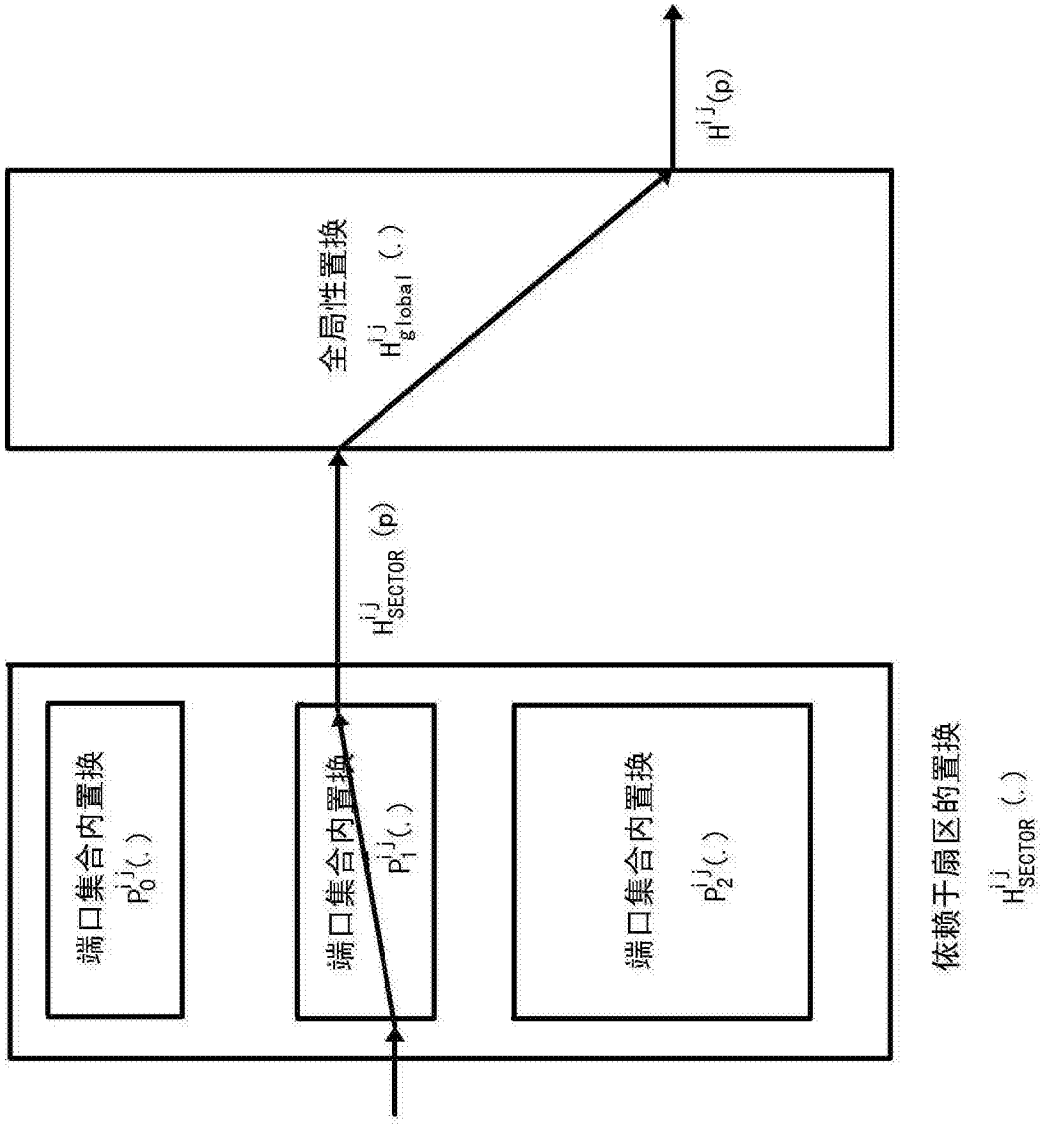

$H^{i,j}(p)$
全局性置换
$H^{i,j}_{global}(.)$
$H^{i,j}_{SECTOR}(p)$
端口集合内置换
$P^{i,j}_0(.)$
端口集合内置换
$P^{i,j}_1(.)$
端口集合内置换
$P^{i,j}_2(.)$
依赖于扇区的置换
$H^{i,j}_{SECTOR}(.)$

图 3

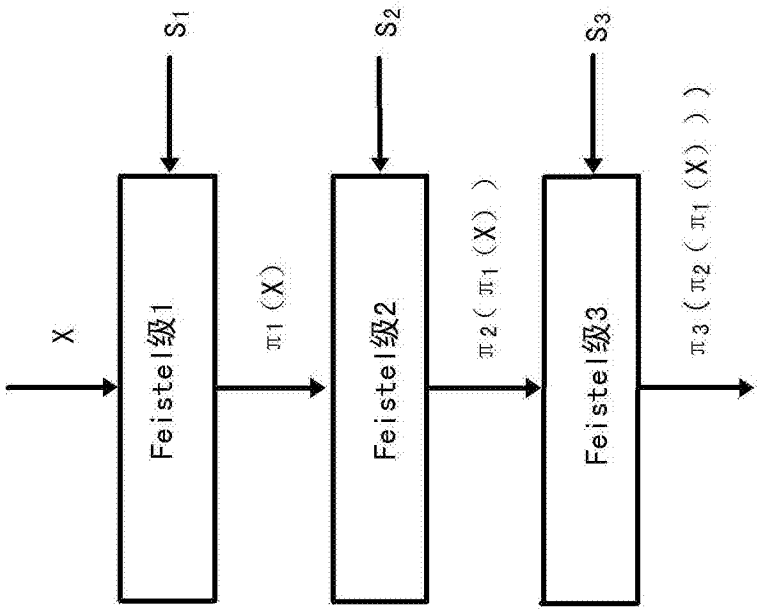

S1
S2
S3
X
Feistel级1
Feistel级2
Feistel级3
π1（X）
π2（π1（X））
π3（π2（π1（X）））

图 4

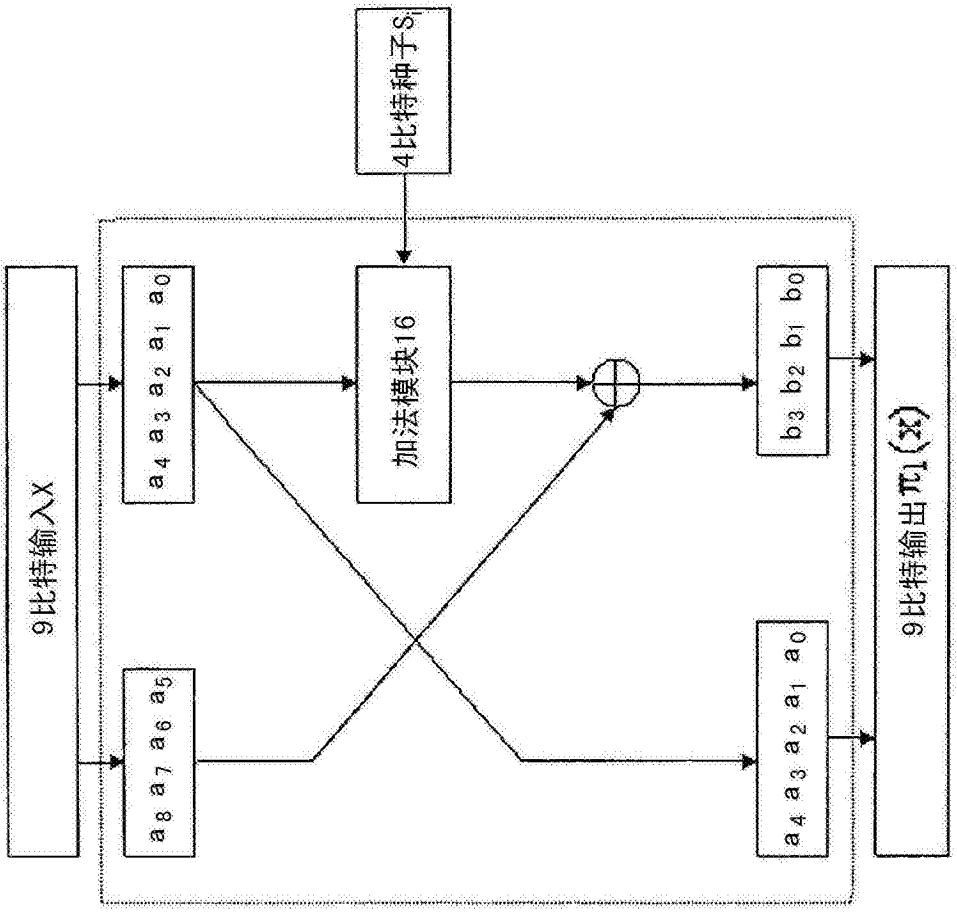

4比特种子Si
9比特输入X
a4 a3 a2 a1 a0
a8 a7 a6 a5
加法模块16
b3 b2 b1 b0
a4 a3 a2 a1 a0
9比特输出π1(x)

图 5

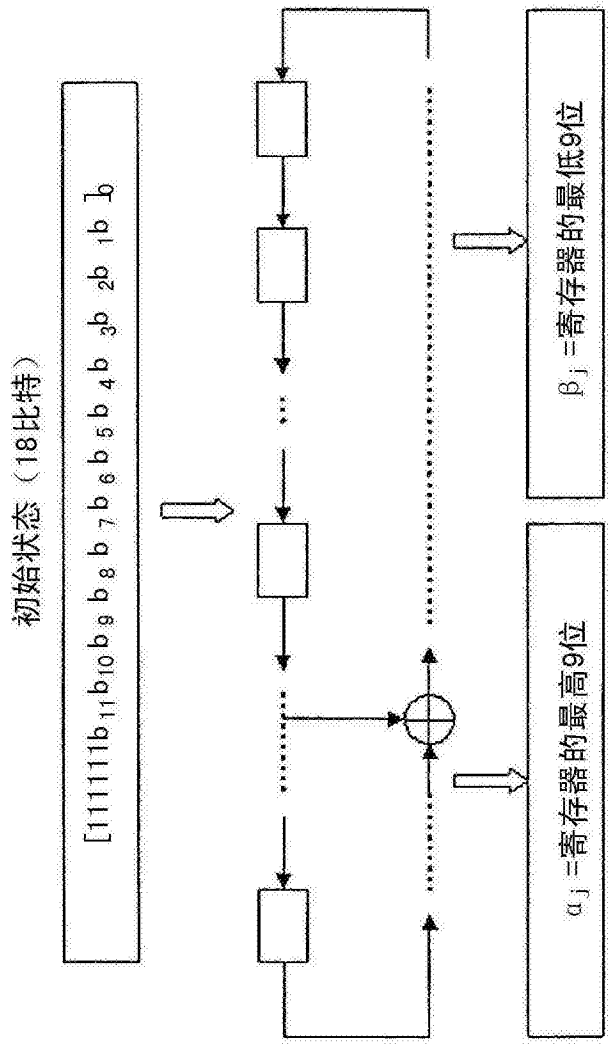

初始状态（18比特）
[111111b11b10b9b8b7b6b5b4b3b2b1b0
βj =寄存器的最低9位
αj =寄存器的最高9位

图 6

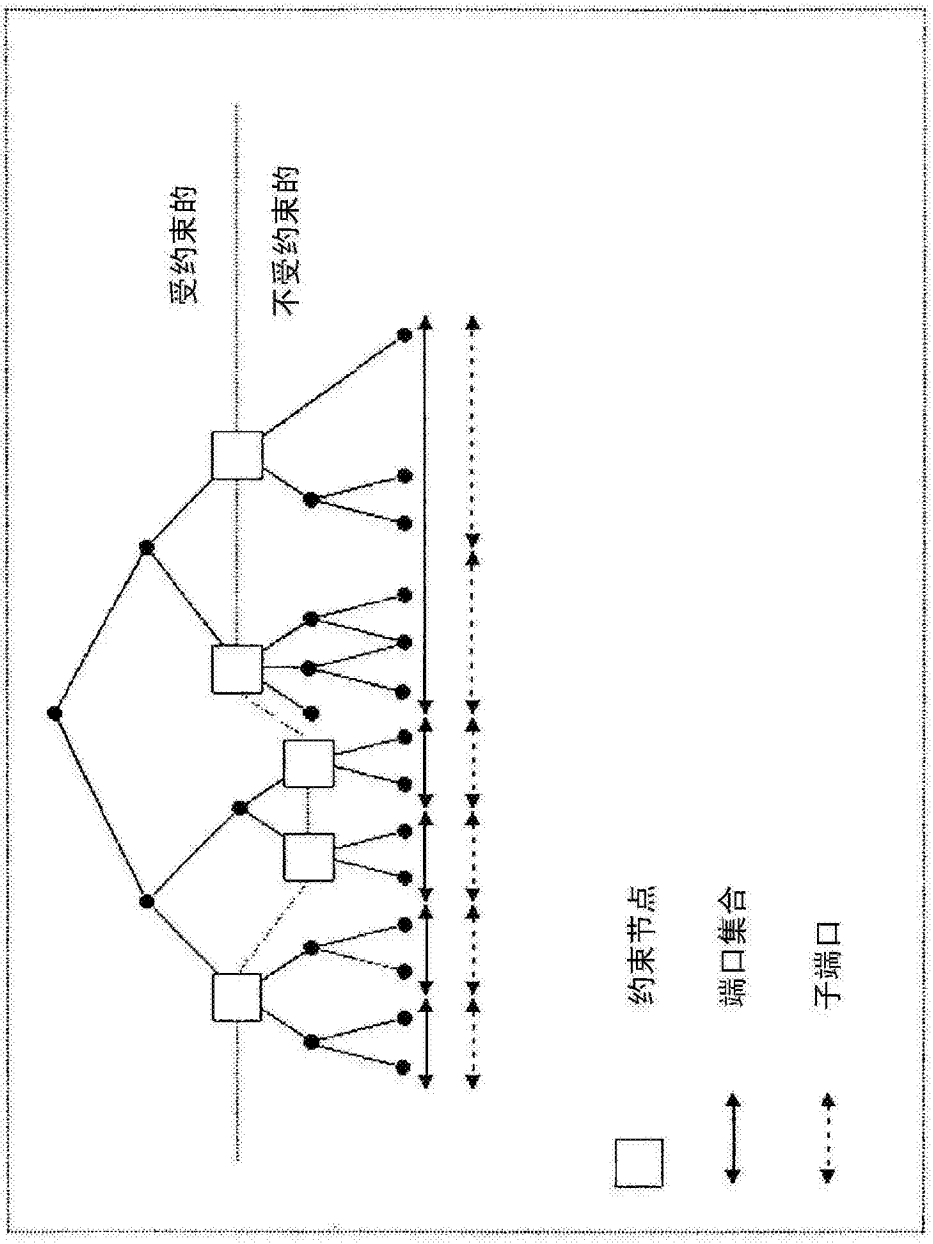


图 7