[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G11B 20/10
G11B 7/00

[12] 发明专利申请公开说明书

[21] 申请号 200510073977.7

[43] 公开日 2005 年 11 月 23 日

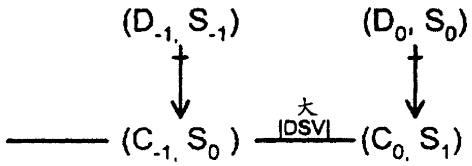
[11] 公开号 CN 1700334A

[22] 申请日 2005.5.19
[21] 申请号 200510073977.7
[30] 优先权
[32] 2004. 5.19 [33] GB [31] 0411163.9
[71] 申请人 麦克罗维西恩欧洲公司
地址 英国伯克郡
[72] 发明人 卡门·L·巴齐尔
[74] 专利代理机构 北京市柳沈律师事务所
代理人 黄小临 王志森
权利要求书 3 页 说明书 15 页 附图 15 页

[54] 发明名称 光盘的复制保护

[57] 摘要

具有大 DSV 绝对值的数据符号的破坏性 DSV(SDSV)序列在光盘的复制保护中非常有用，因为它们可以引起无法纠正的读错误。 但是，在诸如用在 DVD 中的八到十六调制(ESM)的多模式码中能够找到极少量的数据符号的 SDSV 序列。 需要选择用于使用多模式码编码的数据符号，其能够促使编码器产生至少一个破坏性码字序列。 如果对于数据符号的可能的码字具有大 DSV 绝对值并且不存在替代性码字，或者所有的替代性码字都是等价的，或者除一个以外所有的替代项被 RLL 规则排除，则选择该可能的码字。



ISSN1008-4274

1. 一种对用户数据被编码在其上的光盘进行复制保护的方法，所述编码利用了多模式码，并且所述方法包括将所选择的数据符号并入要被编码到所述盘上的用户数据中，以确保至少一个具有大 DSV 绝对值的破坏性码字序列被编码到所述盘上。

2. 如权利要求 1 所述的对光盘进行复制保护的方法，其中，所述破坏性码字序列或者每个破坏性码字序列具有大 DSV 绝对值。

3. 如权利要求 1 或 2 所述的对光盘进行复制保护的方法，其中，所述破坏性码字序列或者每个破坏性码字序列具有偶数个跳变。

4.如任一前述权利要求所述的对光盘进行复制保护的方法，其中，提供了破坏性数据符号序列，当其在特定状态 S 下被编码时将促使编码器输出 S 作为该序列的下一状态。

5.如任一前述权利要求所述的对光盘进行复制保护的方法，其中，所述破坏性序列或每个破坏性序列中的每个码字是对于被并入用户数据中的相应的所选择的数据符号的别无选择的唯一码字。

6.如任一前述权利要求所述的对光盘进行复制保护的方法，其中，所述破坏性序列或每个破坏性序列中的码字的一些是对于被并入用户数据中的相应的所选择的数据符号的两个或更多个替代项之一，但是两个替代项的每个是等价的。

7.如任一前述权利要求所述的对光盘进行复制保护的方法，其中，所述破坏性序列或者每个破坏性序列中的码字的一些是对于被并入用户数据中的相应的所选择的数据符号来说两个或更多个替代项之一，但是除了一个外所有的替代项都被 RLL 规则排除。

8. 一种用户数据被编码在其上的受复制保护的光盘，所述编码利用了多模式码，其中，至少一个具有大 DSV 绝对值的破坏性码字序列被编码到所述盘上，所述破坏性码字序列或每个破坏性码字序列是从被并入用户数据中的所选择的用户符号获得的。

9. 如权利要求 8 所述的受复制保护的光盘，其中，所述破坏性码字序列或者每个破坏性码字序列具有大 DSV 绝对值。

10. 如权利要求 8 或 9 所述的受复制保护的光盘，其中，所述破坏性码

字序列或者每个破坏性码字序列具有偶数个跳变。

11. 如权利要求 8 到 10 的任一个所述的受复制保护的光盘，其中，所述破坏性序列或每个破坏性序列中的每个码字是对于被并入用户数据中的相应的所选择的数据符号的别无选择的唯一码字。

12. 如权利要求 8 到 10 的任一个所述的受复制保护的光盘，其中，所述破坏性序列或每个破坏性序列中的码字的一些是对于被并入用户数据中的相应的所选择的数据符号的两个或更多个替代项之一，但是两个替代项的每个是等价的。

13. 如权利要求 8 到 10 的任一个所述的受复制保护的光盘，其中，所述破坏性序列或者每个破坏性序列中的码字的一些是对于被并入用户数据中的相应的所选择的数据符号来说两个或更多个替代项之一，但是除了一个外所有的替代项都被 RLL 规则排除。

14. 一种利用多模式码对用户数据编码的方法，所述方法包括将所选择的数据符号并入用户数据中，所述数据符号被选择来促使编码器产生至少一个具有大 DSV 绝对值的破坏性码字序列。

15. 如权利要求 14 所述的对用户数据编码的方法，其中，提供了破坏性数据符号序列，当其在特定状态 S 下被编码时将促使编码器输出 S 作为该序列的下一状态。

16. 如权利要求 14 或 15 所述的对用户数据编码的方法，其中，所述破坏性序列或每个破坏性序列中的码字的一些是对于被并入用户数据中的相应的所选择的数据符号的两个或更多个替代项之一，但是两个替代项的每个是等价的。

17. 如权利要求 14 或 15 所述的对用户数据编码的方法，其中，所述破坏性序列或者每个破坏性序列中的码字的一些是对于被并入用户数据中的相应的所选择的数据符号来说两个或更多个替代项之一，但是除了一个外所有的替代项都被 RLL 规则排除。

18. 如权利要求 14 所述的对用户数据编码的方法，其中，所选择的数据符号是通过下述步骤确认的——查看输入数据符号序列的码字；并且确定：是否码字序列具有偶数个跳变，是否码字序列具有与初始状态相同的下一状态，是否不存在替代性码字序列或者所有替代性码字序列都是等价的、或者两个替代性序列之一违反了 RLL 规则，是否码字序列具有大 DSV 绝对值；

以及如果满足所有条件则选择该数据符号以并入用户数据中。

19. 一种选择用于并入要利用多模式码编码的用户数据中的数据符号的方法，所选择的数据符号被选择，使得它们能够促使编码器产生至少一个破坏性码字序列，所述方法包括：查看数据符号的可能的码字，以及如果其码字具有大 DSV 绝对值并且不存在替代性码字或者所有替代性码字等价，或者除一个以外所有替代项都被 RLL 规则排除，则选择该数据符号。

20. 一种选择用于并入要利用多模式码编码的用户数据中的数据符号的方法，所选择的数据符号被选择，使得它们能够促使编码器产生至少一个破坏性码字序列，所述方法包括：查看对于两个或多个数据符号的序列的码字序列，以及如果该码字序列具有大 DSV 绝对值并且不存在替代性码字序列或者所有替代性序列等价，或者除一个以外所有替代项都被 RLL 规则排除，则选择该两个或多个数据符号的序列。

光盘的复制保护

技术领域

本发明涉及一种对光盘进行复制保护(copy protect)的方法和一种受复制保护的光盘。此外，本发明涉及一种对用户数据进行编码的方法和一种选择用于并入用户数据中的数据符号的方法。

背景技术

诸如各种格式的紧密盘(compact discs, CD)和数字多功能盘(digital versatile disc, DVD)的光盘越来越多地用于为许多不同的应用承载信息。编码到光盘上的信息通常是非常有价值的，因而其越来越多地被伪造者复制。此外，可记录 CD 和用于将信息内容从一个盘写到这种可记录盘上的 CD 记录器(CD writer)很容易为国内消费者得到。可记录 DCD 和 DVD 记录器已经很容易得到。这意味着需要用于对光盘进行复制保护的新的、有效的方法。

本申请者已提出了利用具有较差的 DSV 特性的数据模式(data pattern)的各种复制保护技术。例如，在 WO 02/11136 中数据模式被加到 CD 上以提供认证签名(authenticating signature)。选取这些数据模式以引起 DSV 问题。已经发现，当 CD 记录器被用来制造原始盘的拷贝时，其难于写入认证签名。

在 PCT/GB2004/000241 中，通过记录到具有较差的 DSV 特性的盘数据上而将错乱的 dc 内容的区域添加到光盘上。已经发现例如如果带有错乱的 dc 内容的记录数据的区域在大小上严格受限，则在正常播放盘时没有任何问题，但同样对盘复制变得非常困难。

由上可以看出，将具有较差的 DSV 特性的数据模式压印(impress)到光盘上非常有用。

发明内容

本发明试图提供一种通过将具有较差 DSV 特性的破坏性(subversive)数据压印到光盘上来对光盘进行复制保护的方法。

根据本发明第一方面，提供了一种对用户数据被编码在其上的光盘进行

复制保护的方法，所述编码利用了多模式(multimodal)码，并且所述方法包括将所选择的数据符号并入要被编码到光盘上的用户数据中，以确保至少一个具有大 DSV 绝对值的破坏性码字(code word)序列被编码到所述盘上。

如果可以通过简单选择用户数据中的数据符号将破坏性码字序列编码到所述盘上，则其将是非常有用的。需要选取这些数据符号使得它们将促使任何编码器输出破坏性码字序列。

优选地，所述破坏性码字序列或者每个破坏性码字序列具有大 DSV 绝对值。

在一个实施例中，提供了促使产生具有偶数个跳变(transition)的破坏性码字序列的数据符号序列。

优选地，提供了破坏性数据符号序列，当其在特定状态 S 下被编码时将促使编码器输出 S 作为该序列的下一状态。

在一个实施例中，所述破坏性序列或每个破坏性序列中的每个码字是对于被并入用户数据中的相应的所选择的数据符号的别无选择的唯一码字。

另外和/或作为选择地，所述破坏性序列或每个破坏性序列中的码字的一些是对于被并入用户数据中的相应的所选择的数据符号的两个或更多个替代项(alternatives)之一，但是两个替代项的每个是等价的。

另外和/或作为选择地，所述破坏性序列或者每个破坏性序列中的码字的一些是对于被并入用户数据中的相应的所选择的数据符号来说两个或更多个替代项之一，但是除了一个外所有的替代项都被 RLL 规则排除。

如上面所指出的，破坏性码字序列可用于提供认证签名。

另外和/或作为选择地，破坏性码字序列可用于对盘上编码数据的所选区域供给错乱的 dc 内容。

优选地，所述破坏性码字序列或每个破坏性码字序列具有快变化速率的 DSV。

根据本发明另一方面，提供了一种用户数据被编码在其上的受复制保护的光盘，所述编码利用了多模式码，其中，至少一个具有大 DSV 绝对值的破坏性码字序列被编码到所述盘上，所述破坏性码字序列或每个破坏性码字序列是从被并入用户数据中的所选择的数据符号获得的。

优选地，所述破坏性码字序列或者每个破坏性码字序列具有大 DSV 绝对值。

另外和/或作为选择地，所述破坏性码字序列或者每个破坏性码字序列具有偶数个跳变。

另外和/或作为选择地，提供了破坏性数据符号序列，当其在特定状态 S 下被编码时将促使编码器输出 S 作为该序列的下一状态。

另外和/或作为选择地，所述破坏性序列或每个破坏性序列中的每个码字是对于被并入用户数据中的相应的所选择的数据符号的别无选择的唯一码字。

在一个实施例中，所述破坏性序列或每个破坏性序列中的码字的一些是对于被并入用户数据中的相应的所选择的数据符号的两个或更多个替代项之一，但是两个替代项的每个是等价的。

另外和/或作为选择地，所述破坏性序列或者每个破坏性序列中的码字的一些是对于被并入用户数据中的相应的所选择的数据符号来说两个或更多个替代项之一，但是除了一个外所有的替代项都被 RLL 规则排除。

所述破坏性码字序列或者每个破坏性码字序列可用于提供认证签名。

另外和/或作为选择地，所述破坏性码字序列或者每个破坏性码字序列用于对盘上编码数据的所选区域供给错乱的 dc 内容。

优选地，所述破坏性码字序列或者每个破坏性码字序列具有快变化速率的 DSV。

本发明还提供了一种利用多模式码对用户数据编码的方法，所述方法包括将所选择的数据符号并入用户数据中，该数据符号被选择来促使编码器产生至少一个具有大 DSV 绝对值的破坏性码字序列。

当处理诸如用在 CD 中的 EFM 调制的非多模式码时，相对直接地选取破坏性码字序列然后将该序列解码成用于并入用户数据中的数据符号。但是，为了对 DVD 采取同样的操作，将需要浩大的计算时间。

在一个实施例中，所述破坏性码字序列或者每个破坏性码字序列具有大 DSV 绝对值。

优选地，所述破坏性码字序列或者每个破坏性码字序列具有偶数个跳变。

另外和/或作为选择地，一种破坏性数据符号序列，当其在特定状态 S 下被编码时将促使编码器输出 S 作为该序列的下一状态。

另外和/或作为选择地，所述破坏性序列或每个破坏性序列中的每个码字是对于被并入用户数据中的相应的所选择的数据符号的别无选择的唯一码

字。

另外和/或作为选择地，所述破坏性序列或每个破坏性序列中的码字的一些是对于被并入用户数据中的相应的所选择的数据符号的两个或更多个替代项之一，但是两个替代项的每个是等价的。

另外和/或作为选择地，所述破坏性序列或者每个破坏性序列中的码字的一些是对于被并入用户数据中的相应的所选择的数据符号来说两个或更多个替代项之一，但是除了一个外所有的替代项都被 RLL 规则排除。

在优选实施例中，每个所选择的数据符号被确认(identify)为具有大 DSV 绝对值码字的数据符号，其中或者不存在替代性码字或者所有可能的替代项都具有大 DSV 绝对值。

因此，在实施例中，通过下述步骤确认所选择的数据符号——查看输入数据符号序列的码字序列；并且确定(establish)：是否码字序列具有偶数个跳变，是否码字序列具有与初始状态相同的下一状态，是否不存在替代性码字序列或者所有替代性码字序列都是等价的、或者两个或多个替代性序列之一违反了 RLL 规则，是否码字序列具有大 DSV 绝对值；以及如果满足所有条件则选择该数据符号以并入用户数据中。

本发明还提供了一种选择用于并入要利用多模式码编码的用户数据中的数据符号的方法，所选择的数据符号被选择，使得它们能够促使编码器产生至少一个破坏性码字序列，所述方法包括：查看数据符号的可能的码字，以及如果其码字具有大 DSV 绝对值并且不存在替代性码字或者所有替代性码字等价，或者两个可供选择序列之一违反了 RLL 规则，则选择该数据符号。

根据本发明另一方面，提供了一种选择用于并入要利用多模式码编码的用户数据中的数据符号的方法，所选择的数据符号被选择，使得它们能够促使编码器产生至少一个破坏性码字序列，所述方法包括：查看对于两个或多个数据符号的序列的码字序列，以及如果该码字序列具有大 DSV 绝对值并且不存在替代性码字序列或者所有替代性序列等价，或者两个可供选择序列之一违反了 RLL 规则，则选择该两个或多个数据符号的序列。

优选地，该方法还包括选择其中码字序列具有偶数个跳变的数字符号序列。

该方法还可以包括选择其中码字序列的下一状态与其初始状态相同的数据符号序列。

附图说明

下文中将参考附图利用示例描述本发明，在附图中：

图 1 图示了对数据符号编码以产生码字，

图 2 图示了对数据符号序列进行编码，

图 3 图示了码字的状态，

图 4 图示了多模式码提供的编码选项，

图 5 图示了输入数据符号的两个可能的输出码字，

图 6 图示了 ESM 编码可得的码字，

图 7 示出了对于其中编码器试图最小化 DSV 绝对值的到光盘的应用的编码，

图 8 示出了在对于到 DVD 的应用准备数据时遇到的数据层次(data level)，

图 9 图示了非多模式码的编码和解码，

图 10 图示了多模式码的编码和解码，

图 11 图示了码字的特性，

图 12 示出了一对数据符号和它们的码字的三种可能情形，

图 13 图示了步骤 1 之后图 12 的三种情形，

图 14 图示了步骤 2 之后图 13 的三种情形，

图 15 图示了图 14 的三种情形的可能子情形，

图 16 示出了三种子情形(3.1)、(3.2)和(3.3)，

图 17 示出了将促使编码器选择具有大 DSV 绝对值的码字的数据符号序列，

图 18 示出了在步骤 1 和步骤 2 之后获得的数据符号序列的示例，其将促使编码器输出 SDSV 序列，而其不是 SDSV 模式，以及

图 19 示出了数据符号的 SDSV 模式的示例。

具体实施方式

多模式码

多模式码是基于状态机的游程长度限制(Run Length Limitation, RLL)码，其中最优符号选项不仅依赖于编码器状态和要编码的数据而且依赖于诸

如 DSV 的某些非局部特性。用在 DVD 盘中的八到十六调制(Eight-to-Sixteen Modulation, ESM 或 EFM+)构成这样的码的示例。

基于状态机的 RLL 码的基本结构如下。如果 k 和 d 分别是在编码序列中所允许的连续的零的最小数目和最大数目，则认为该码是 RLL(k,d)码。

如图 1 所示，给定输入数据符号 D(i)和状态 S(i)，输出码字

$$C(i)=C(D(i),\ S(i))$$

将和下一状态

$$S(i+1)=S(D(i),S(i)),$$

一起返回，其中 C(,)是输出码字函数和 S(,)是下一状态函数。认为码字 C(i)处于状态 S(i)。假定给出了输入数据符号序列

$$\{D(0), D(1),\ldots, D(n)\}$$

和初始状态 S(0)。对每一组对(pair)(D(i), S(i))，将生成新的组对(C(i), S(i+1))，如图 2 所示，其中

$$C(i)=C(D(i), S(i));$$

$$S(i+1)=S(D(i), S(i)).$$

下一状态 S(i+1)是其中将对数据符号 D(i+1)进行编码的状态。那么输出码字序列将是

$$\{C(0), C(1), \ldots, C(n)\},$$

其中 C(0)处于状态 S(0)，C(1)处于状态 S(1)，…，C(n)处于状态 S(n)。输出码字形成了满足 RLL(k, d)规则的比特序列。

码字 C 的状态可基本由其 RLL 特性来定义。更确切地说，其可根据 C 可以跟随的码字的种类来定义而不违反 RLL 规则。例如，考虑不带尾随零(trailing zero)的码字的种类。则状态 S_1 可定义为其中所有的码字具有至少 k 个前导零(leading zero)的状态。给出状态 S_1 的该定义，则不带尾随零的码字后可跟随处于状态 S1 的任何码字。因而，状态 S_1 可被设置为所有不带尾随零的码字的下一状态。类似地，考虑具有 d 个尾随零的码字的种类，并且定义状态 S_2 为其中所有的码字没有前导零的状态，从而可将状态 S_2 设置为所有具有 d 个尾随零的码字的下一状态。图 3 示出了作为 RLL(2,10)码的 ESM 的一些示例。在 ESM 中，状态 1 被定义为其中所有的码字具有至少两个前导零的类。因而码字 0010000000001001 的下一状态被设置为状态 1。

类似地，状态 4 被定义为其中所有的码字具有至多一个前导零的类。由

于在 ESM 中不存在具有多于 9 个尾随零的码字，所以任何具有多于 2 个尾随零的 ESM 码字后可跟随处于状态 4 的码字并可将下一状态设置为状态 4。

多模式码以输入数据符号 40 可以被编码的形式提供了选项，如图 4 所示，其中提供了两个替代项 42 和 44。对数据符号的每个输入序列，通常存在许多不同可能的码字输出序列。编码器将根据编码序列的某些非局部特性诸如 DSV 来在所有可能的选项之中选择一个输出序列。

例如，图 5 示出了数据符号输入序列{D(i-1), D(i)}具有两个可能的输出序列，相应于路径 A 50 的序列{C(i-1), C(i)}和相应于路径 B 52 的序列{C(i-1), C'(i)}。因而编码器可在对于输入序列{D(i-1), D(i)}的两个替代性输出之间进行选择。如果编码器被设计成最小化 DSV 的绝对值(|DSV|)，则其很明显将选择路径 A。

ESM 是将 8 比特输入数据符号转换成 16 信道比特码字的 4 状态多模式码。该转换是根据两个查找转换表即主表(Main Table)和替换表(Substitution Table)进行的。对于每个状态及对于每个输入数据符号，主表包含有相应的 ESM 码字的列表。替换表包含对包括在 0,…,87 范围内的数据符号的替代性编码。因而，给出在 0,…,87 范围内的数据符号 D(i)和状态 S(i)，存在着两个替代性输出 C(i), S(i+1)和 C'(i), S(i+1)，一个来自主表，另一个来自替换表。对于要在状态 1 或状态 4 被编码的处于 88,…,255 范围内的数据符号，也可以具有替代性输出：假定满足 RLL 规则，则要在状态 1 编码的数据符号 88,…,255 也可以在状态 4 编码，类似地，要在状态 4 编码的数据符号 88,…,255 也可以在状态 1 编码。对要在状态 2 或状态 3 编码的处于 88,…,255 范围内的数据符号不存在替代性编码。图 6 示出了数据符号的可得输出。

用于执行转换的表和方法以这样的方式安排，从而可使码字的输出序列的 DSV 的绝对值(|DSV|)最小化，如图 7 中所图示的那样。因而“智能编码器”即，被设计成在许多输出选项之中选择最优选项的编码器，通常将能够有效地最小化|DSV|。但是，存在即使智能编码器也被促使输出具有相对大的|DSV|值的码字序列的情况，或者因为不存在该序列的可得的替代项，或者因为可能替代项都将导致大|DSV|值。

如果一个码字序列当被从光盘读出时能够引起无法纠正的读错误，则认为该码字序列是破坏性序列。如果当供给数据符号输入序列时编码器被促使输出破坏性码字序列，则认为该输入数据符号序列是破坏性序列。

公知地，具有大|DSV|的编码序列可引起无法纠正的读错误。在这种情况下，来讨论一下破坏性 DSV(SDSV)序列。

获取 SDSV 序列的问题

输入数据符号的 SDSV 序列对基于破坏性数据的复制保护技术非常有用，这是因为它们允许通过独有地工作在用户数据 2 层而不是在物理扇区(physical sector)4 层因而在将用户数据写到 DLT 带(DLT tape)上之前，在盘上创建不可读的数据。图 8 中示出了将用户数据施加到例如 DVD 的光盘时的数据层(data level)。换言之，在用户数据 2 中插入数据符号的 SDSV 序列将促使给定的用于 EFM+调制(EFM Plus Modulation)的编码器输出包含码字的 SDSV 序列的物理扇区 4。

但是，在诸如 ESM 的多模式码中，能够发现能够调节(tweak)智能编码器的极少量的数据符号的 SDSV 序列。这使得考虑到数据符号的所有可能序列和作为智能解码器的输出的其相应的码字的编码序列的|DSV|以寻找数据符号的 SDSV 序列的穷尽式方法很不可行。

另一种替代性方法可以由下述步骤组成，即从码字的 SDSV 序列开始，并且使用解码器将这些序列解码成数据符号序列。如图 9 和图 10 所示，虽然这对于诸如 EFM 调制的非多模式码可能是一种可行途径，但在多模式码的情况下，情形更加复杂。图 9 示出了非多模式码的情况：将码字序列 8 解码成数据符号序列 10；然后对数据符号序列编码以给出等于码字序列 8 的码字输出序列 12。图 10 示出了类似处理但处于多模式码的情况：在这种情况下，码字的输出编码序列 12'不一定等于码字序列 12。因为构造 ESM 转换表码的方式以及因为被编码器采用来进行转换的算法，所以在大多数情况下，给定码字的 SDSV 序列 8，从该码字的 SDSV 序列解码得到的数据符号序列 10 将还具有码字的替代性的非 SDSV 编码序列，转换算法更倾向于该非 SDSV 序列而不是 SDSV 序列。因而，给定数据符号输入序列 10，智能编码器将输出码字的非 SDSV 序列 12'而不是 SDSV 序列 8。

由此得出结论，在多模式码中对 SDSV 序列的任何穷尽式搜索是计算量浩大的。

如何获得 SDSV 序列

需要找到一种方法，其能够确定促使破坏性序列(forced subversive sequences)，即能够促使编码器输出破坏性码字序列的数据符号序列。具体地

说，需要生成用于 ESM 调制的 SDSV 模式(pattern)，即数据符号序列 $\{D_0,\cdots,D_r\}$(加上初始状态)，使得相应的码字编码序列$\{C_0,\cdots,C_r\}$具有“大”|DSV|并且使得从 DSV 的观点看可以有效地重复它们如需要的那么多次。

更精确地，如果具有初始状态 S_0 的数据符号序列$\{D_0,\cdots,D_r\}$当被重复假定 t 次时

$$\{\underbrace{D_0,\cdots,D_r}_{1},\underbrace{D_0,\cdots,D_r}_{2},\ \cdots,\underbrace{D_0,\cdots,D_r}_{t}\}$$

将促使给定的编码器输出下述码字序列

$$\{\underbrace{C_0,\cdots,C_r}_{1},\underbrace{C_0,\cdots,C_r}_{2},\ \cdots,\underbrace{C_0,\cdots,C_r}_{t}\}$$

它的|DSV|是

$$|DSV\{\underbrace{C_0,\cdots,C_r}_{1},\underbrace{C_0,\cdots,C_r}_{2},\ \cdots,\underbrace{C_0,\cdots,C_r}_{t}\}|=t\times|DSV(C_0,\cdots,C_r)|$$

则具有初始状态 S_0 的数据符号序列$\{D_0,\cdots,D_r\}$是 SDSV 模式。

该方法是具体参考 ESM 描述的。但是，所概述的方法可以与除了 ESM 的多模式 RLL 码一起使用。

优选地，该方法将提供数据符号模式的列表，其促使得到能够在被进行 ESM 调制时导致 SDSV 的大|DSV|。

如果输入数据在 ESM 之前经受诸如扰乱(scramble)的某种操作，则当写 SDSV 序列时必须考虑该操作，从而在该操作后这些序列将导致促使 SDSV 序列。

现在将描述一种用于生成促使 SDSV 序列，具体地说即 SDSV 模式的方法。

如图 11 所示，对每个码字，需要考虑下列特性。

- DSV;
- 跳变的数目(即，码字包含的 1 的数目);
- 状态;

- 下一状态。

给定码字 C，将使用下面的记号：

- DSV(C)表示带有正负号的 C 的 DSC;
- |DSV(C)|表示 C 的|DSV|;
- Transitions(C)表示 C 的跳变的数目；
- State(C)表示其中对 C 编码的状态;
- NextState(C)表示 C 的下一状态。

依照惯例，如图 11 所示那样计算码字的 DSV。

注意，上面的概念也适用于任何比特序列，从而，具体地适用于码字序列。因而，当考虑码字序列时上面的记号也适用。

认为两个组对 (C, S)和($C^{'}$, $S^{'}$)(或两个序列{(C_j, S_j)}和{($C_j^{'}$, $S_j^{'}$)})组对是等价的，当且仅当

a) DSV(C) DSV($C^{'}$)>=0(即，DSV(C)和 DSV($C^{'}$)具有同样的正负号或者两者之一为零);

b) Transitions(C)和 Transitions($C^{'}$)奇偶性相同(即，它们均为偶数或者均为奇数);

c) S=$S^{'}$;

d) |DSV(C)|和|DSV($C^{'}$)| “近似相等(almost equal)”。

如果|DSV($C^{'}$)| =|DSV(C)|+L，其中 L 是有正负之分的整数则|DSV(C)|和|DSV($C^{'}$)| “近似相等”。|L|越小，“近似相等” 的定义限定越严格。

如果(C, S)和($C^{'}$, $S^{'}$)等价，则写成

$$(C, S) \sim (C^{'}, S^{'});$$

如果它们不等价，则写成

$$\neg(C, S) \sim (C^{'}, S^{'})。$$

注意，具有给定初始状态 S_0 的数据符号序列{D_0,...,D_r}是 SDSV 模式，条件是相应的码字编码序列{ C_0,...,C_r }满足下面的条件：

a) 序列{C_0,...,C_r}的跳变数目是偶数;

b) 当以初始状态 S_0 编码时序列{D_0,...,D_r}的下一状态是 S_0;

c) |DSV{(C_0,...,C_r)}| “大”

d) 或者不存在替代性编码序列，或者如果存在替代性编码序列{ $C_0^{'}$,...,$C_r^{'}$ }的话，则其 “等价于” {C_0,...,C_r}，否则其将因为{C_r,$C_0^{'}$}

违反了 RLL 规则而被排除。

假设 m_0 是所有 ESM 码字中的最大|DSV|值。则考虑下列码字，其具有

$$|DSV|=m_0-2i$$

对 i=0, ...,M, 其中 M 是满足 0<=M <=m_0/2 的整数。M 的值取决于所需的 SDSV 序列必须有多强(how strong)。

注意，偶数长的比特序列的 DSV 值永远是偶数。

在下文中，假定，给定转换表，编码算法将关于最小化|DSV|尽可能地有效(effective)。由于通常并不如此，所以可以使下面描述的方法适应于所使用的特定编码算法，以开拓其弱点。

方法概述

对 i=0, ...,M，其中 0<=M <=m_0/2，假设 C_0 是使得下式成立的码字，

$$|DSV(C_0)| = m_0-2i$$

假设 D_0 和 S_0 分别是使得下式成立的数据符号和状态，

$$C_0=C(D_0, S_0)$$

组对(D_0, S_0)不一定是唯一确定的。可能存在不同的组对(D_0, S_0)和(D_0', S_0')使得 $C(D_0, S_0)= C(D_0', S_0)$。

步骤 1

假设(D_{-1}, S_{-1})使得 $S(D_{-1}, S_{-1}) = S_0$ 并设 $C_{-1}=C(D_{-1}, S_{-1})$。如果$|DSV(C_{-1}, C_0)|$“小”，则丢弃组对(D_{-1}, S_{-1})然后检查另一适当的组对(D_{-1}, S_{-1})。

当写下代码 C 使得 C=C(D, S)而没有任何进一步说明时，则意味着 C 是相应于(D, S)的默认的编码码字，即图 6 中的选项 A。

如果$|DSV(C_{-1}, C_0)| < |DSV(C_0)|+T$，则$|DSV(C_{-1}, C_0)|$“小”，其中 T 是使得 0<=T<=$m_0$ 的参数。因而如果$|DSV(C_{-1}, C_0)| >= |DSV(C_0)|+T$ 则$|DSV(C_{-1}, C_0)|$“大”。显然 T 越大，SDSV 序列将越强，如果发现了任何 SDSV 的话。

假定$|DSV(C_{-1}, C_0)|$“大”。则有如图 6 所示的下面的情况之一。

1) D_0 在 0,…,87 的范围内;

2) D_0 在 88,…,255 的范围内并且 S_0 等于状态 1 或状态 4;

3) D_0 在 88,…,255 的范围内并且 S_0 等于状态 2 或状态 3;

在第一种情况下，(C_0, S_1)的替代性组对(C_0', S_1')将永远存在。在第二种情况下中，替代性组对(C_0', S_1')可能存在。在第三种情况下不存在替代。

考虑图 12 中列出的这三种情况。

情况(1)

请参考图 12，情况(1)。如果$\neg(C_0', S_1') \sim (C_0, S_1)$，则丢弃$(D_0, S_0)$并且找到另一适当的组对$(D_0, S_0)$。现在假定$(C_0', S_1') \sim (C_0, S_1)$，如图 13，情况(1)中所示。则可从下面的步骤 2 继续进行。

情况(2)

请参考图 12，情况(2)。如果$\{C_{-1}, C_0'\}$并不违反 RLL 规则，并且$(C_0', S_1') \sim (C_0, S_1)$，则如上面情况(1)中描述的那样继续进行。如果$\{C_{-1}, C_0'\}$不违反 RLL 规则，但$\neg(C_0', S_1') \sim \neg (C_0, S_1)$，则丢弃$(D_{-1}, S_{-1})$，找到另一适当的组对$(D_{-1}, S_{-1})$使得$S(D_{-1}, S_{-1})=S_0$，并从上面的步骤 1 继续进行。如果，最终，$\{C_{-1}, C_0'\}$确实违反了 RLL 规则，则处于图 12，情况(3)的情形并且可以如下面的情况(3)那样继续进行。

情况(3)

处于图 12，情况(3)所示的情形中，可以从下面的步骤 2 继续进行。

步骤 2

现在处于图 13 中所示的三种情况之一，其中路径 P 的任何替代性路径实际上等价于路径 P。由此得出结论，并不限制忽略任何替代性路径并且假定处于图 13，情况(3)所示的情形中。

有如图 14 所示的三种可能的子情况。

1. D_{-1}在 0,…,87 的范围内；
2. D_{-1}在 88,…,255 的范围内并且S_{-1}等于状态 1 或状态 4；
3. D_{-1}在 88,…,255 的范围内并且S_{-1}等于状态 2 或状态 3；

情况(3.1)

为简单起见，如果$\neg(C_{-1}', S_0') \sim (C_{-1}, S_0)$，则丢弃$(D_{-1}, S_{-1})$并且找到另一适当的组对$(D_{-1}, S_{-1})$。注意，实际上，不必有

$$(C_{-1}', S_0') \sim (C_{-1}, S_0)$$

因为检查下面的就足够了

i. $(C_0'', S_1'') \sim (C_0''', S_1''')$且

ii. $\{(C_{-1}', S_0'), (C_0'', S_1'')\} \sim \{(C_{-1}, S_0), (C_0, S_1)\}$

并仅当如图 15，情况(3.1)中所示不满足这些条件之一才丢弃(D_{-1}, S_{-1})。

现在假定处于图 16，情况(3.1)中所示的情形，其中路径 P 的任何替代性路径与其等价。注意C_0''和C_0可能相等也可能不相等，这同样适用于C_0'''

和 C_0'。

情况(3.2)

为简单起见，如果 $\neg(C_{-1}', S_0') \sim (C_{-1}, S_0)$，则丢弃$(D_{-1}, S_{-1})$并且找到另一适当的组对$(D_{-1}, S_{-1})$。实际上，如图 15，情况(3.2)(a)所示，如果(C_{-1}', S_0')和(C_{-1}, S_0)不等价，仍可以找到适当的组对(C_{-2}, S_{-1})使得序列$\{C_{-2}, C_{-1}'\}$违反了 RLL 规则。或者，如图 15，情况(3.2)(b)所示，可以检查上面的条件 i.和 ii.。

因而，可以假定处于图 16，情况(3.2)所示的情形，其中路径 P 的任何替代性路径都与其等价。

情况(3.3)

图 16，情况(3.3)描述了这种情况。

现假定从情况(3.1)、(3.2)或(3.3)的任一种已发现具有初始状态 S_{-1} 的序列$\{D_{-1},D_0\}$，如图 17 所示(仅示出了一个路径，因为任何其它替代性路径都等价于所示出的路径)。如果具有初始状态 S_{-1} 的$\{D_{-1},D_0\}$是 SDSV 模式(根据上面的定义)，则完成。如果不是，则可从上面的步骤 1 继续进行，其中，不是考虑组对(D_{-1}, S_{-1})使得 $S(D_{-1}, S_{-1})=S_0$，现在将考虑组对(D_{-2}, S_{-2})使得 $S(D_{-2}, S_{-2})=S_{-1}$，并且不是考虑序列$\{C_{-1}, C_0\}$，将考虑序列$\{C_{-2}, C_{-1}, C_0\}$。

相反，如果没有发现任何适当的序列$\{D_{-1},D_0\}$，则将检查具有需要的|DSV|值的另一码字 C_0 并从步骤 1 重新开始。一旦穷尽了对于该特定|DSV|值的所有可能情况，则将 i 的值加 1。

将考虑越来越长的序列$\{C_{-n},\ldots,C_{-1},C_0\}$。显然，当 n 达到最大选定长度时，可以输出数据符号$\{D_{-n},\ldots,D_{-1},D_0\}$的相应 SDSV 序列{其不一定是 SDSV 模式}。

示例

假定正考虑具有|DSV|=4 的码字。假定从 ESM 转换表选择了码字

$$C_0=1001001000000100,$$

其具有等于-4 的 DVS。从该表可得 D_0=98 且 S_0=状态 3 使得 $C_0=C(D_0, S_0)$。现在考虑使得 $S(D_{-1}, S_{-1})=S_0$=状态 3 的所有组对(D_{-1}, S_{-1})。假定在这些组对之中已选取 D_{-1}=88, S_{-1}=状态 2。则有

$$C(D_{-1}, S_{-1})=0001000100010000$$

现在，$DSV(C_{-1},C_0)$=+2。但是然后丢弃了该组对(D_{-1}, S_{-1})=(88,状态 2)因为

$|DSV(C_{-1,},C_0)|$“小”，由于

$$|DSV(C_{-1,},C_0)|=2 < |DSV(C_0)|=4。$$

因而，考虑使得 $S(D_{-1}, S_{-1})$=状态 3 的另一组对(D_{-1}, S_{-1})，比如

$$(D_{-1}, S_{-1})=(131, 状态\ 3)。$$

在这种情况下，有

$$C(D_{-1}, S_{-1})=1001001000000100$$

且 $DSV(C_{-1}, C_0)=-8$。因而$|DSV(C_{-1,},C_0)|$足够“大”，因为

$$|DSV(C_{-1,},C_0)| =8 >= |DSV(C_0)|+4$$

注意，D_0在 88,…,255 的范围内，且 S_0=状态 3，同样 D_{-1}在 88,…,255 的范围内，且 S_{-1}=状态 3。图 18 图示了本情形：发现了具有初始状态 S_{-1}的数据符号序列$\{D_{-1}, D_0\}=\{131, 98\}$使得相应的码字序列$\{C_{-1}, C_0\}$具有大|DSV|。现在可以验证具有初始状态 S_{-1} 的$\{D_{-1}, D_0\}$是否是 SDSV 模式。满足了用于定义 SDSV 模式的条件 a), c)和 d)，因为：

a) $Transitions(C_{-1}, C_0)=8$;

b) $|DSV(C_{-1}, C_0)|$大;

c) 不存在替代性的编码序列。

但是 $NextState(D_{-1}, D_0)$=状态 2，其不等于 S_{-1}=状态 3。因而具有初始状态 S_{-1}的(D_{-1},D_0)不是 SDSV 模式。

因而现在寻找使得 $S(D_{-2}, S_{-2})=S_{-1}$=状态 3 的组对(D_{-2}, S_{-2})。故假设

$$(D_{-2,} S_{-2})=(161, 状态\ 2)。$$

则有 $C_{-2}=C(D_{-2}, S_{-2})=0100000000010000$。然后 $DSV(C_{-2}, C_{-1}, C_0)=-12$ 因而 $|DSV(C_{-2}, C_{-1}, C_0)|$“大”，因为

$$|DSV(C_{-2}, C_{-1}, C_0)|=12 >= |DSV(C_{-1}, C_0)|+4。$$

图 19 图示了本情形，注意，同样 D_{-2}处于范围 88,…,255 且状态 S_{-2}=状态 2 因而不存在要考虑的替代性码字 C_{-2}'。

具有初始状态 S_{-2}=状态 2 的数据符号序列$\{D_{-2}, D_{-1}, D_0\}$是 SDSV 模式。确实，满足了用于定义 SDSV 模式的所有条件，因为：

a) $Transitions(C_{-2}, C_{-1}, C_0)=10$;

b) $NextState(D_{-2}, D_{-1}, D_0)$=状态 $2=S_{-2}$;

c) $|DSV(C_{-2}, C_{-1}, C_0)|$大;

d) 不存在替代性的编码序列。

由此得出结论，可有效地（从|DSV|视点看）重复模式(D_{-2}, D_{-1}, D_0)需要的那么多次，假定初始状态是状态 2。更确切地说，数据符号序列

$$\{D_{-2}, D_{-1}, D_0, D_{-2}, D_{-1}, D_0, D_{-1}, D_0, D_{-2}, \ldots\}$$

将促使 ESM 编码器输出当 n 是序列长度时其|DSV|等于 4*n 的码字序列。

SDSV 模式

一旦发现了许多 SDSV 模式，可以画出列出了这些模式及它们的诸如初始状态、DSV 值的特性的表，如下面所示，

模式	初始状态	DSV	每个符号的|DSV|
$\{A_0, A_1\}$	S_0	+8	4
$\{B_0, B_1\}$	R_0	-8	4
$\{C_0, C_1, C_2\}$	S_0	+12	4
$\{D_0, D_1, D_2\}$	R_0	-12	4
…	…	…	…

给出表中的这些数据，可以选择适当的模式并且将它们组合起来以形成更长的 SDSV 模式。这对生成尽可能看起来随机(random-looking)的 SDSV 序列也是有用的。例如，在上面的表中，第一和第三模式具有相同的初始状态，因而，由 SDSV 模式的定义具有相同的下一状态。因而可以构建具有初始状态 S_0 和 DSV=20 的 SDSV 模式$\{A_0, A_1, C_0, C_1, C_2\}$。

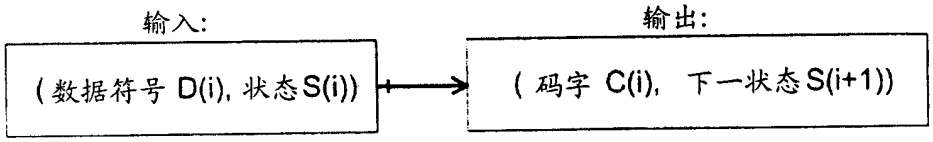

输入:
输出:
(数据符号 D(i), 状态S(i))
(码字 C(i), 下一状态S(i+1))

图 1

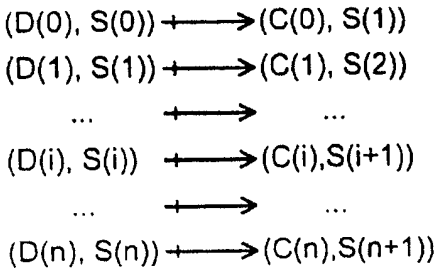

(D(0), S(0)) ⟶ (C(0), S(1))
(D(1), S(1)) ⟶ (C(1), S(2))
... ⟶ ...
(D(i), S(i)) ⟶ (C(i),S(i+1))
... ⟶ ...
(D(n), S(n)) ⟶ (C(n),S(n+1))

图 2

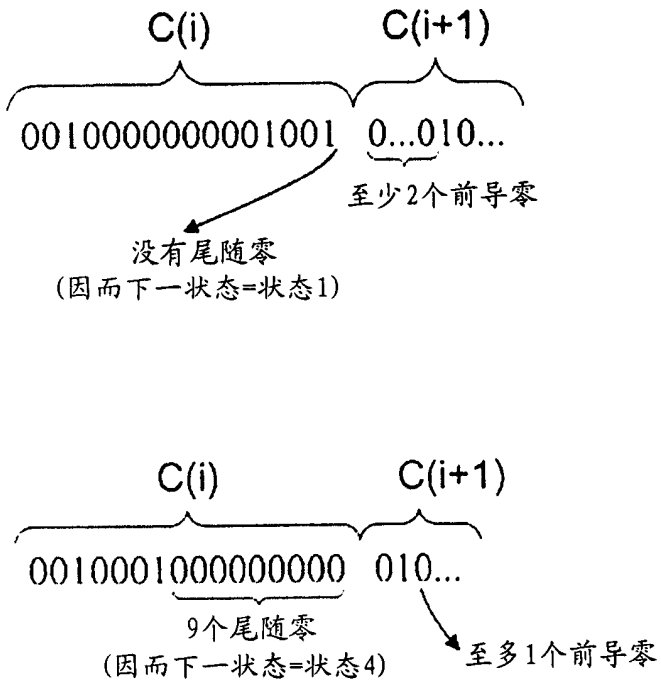

C(i)
C(i+1)
0010000000001001
0...010...
至少2个前导零
没有尾随零
(因而下一状态=状态1)
C(i)
C(i+1)
0010001000000000
010...
9个尾随零
(因而下一状态=状态4)
至多1个前导零

图 3

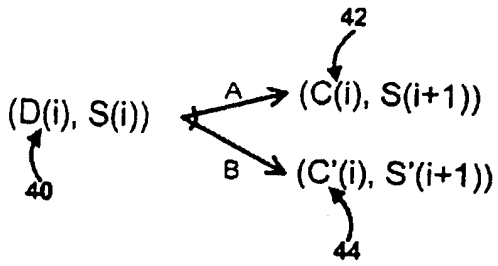

42
(D(i), S(i))
A
(C(i), S(i+1))
B
(C'(i), S'(i+1))
40
44

图 4

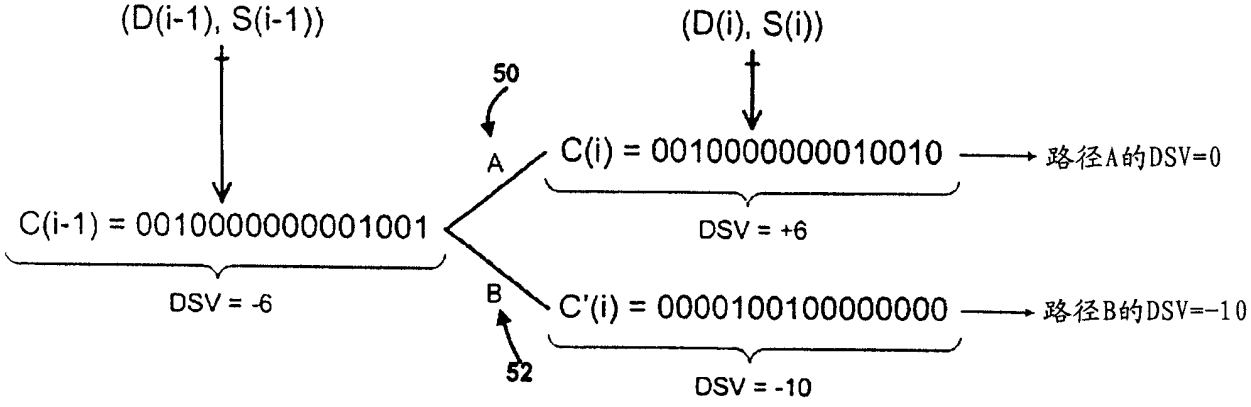

(D(i-1), S(i-1))
(D(i), S(i))
50
A
C(i) = 0010000000010010
路径A的DSV=0
C(i-1) = 0010000000001001
DSV = +6
DSV = -6
B
C'(i) = 0000100100000000
路径B的DSV=-10
52
DSV = -10

图 5

1) D(i) = 0, ..., 87
(D(i), S(i))
A
(C(i), S(i+1)) 其中 C(i) = C(D(i), S(i)) S(i+1) = S(D(i), S(i)) 来自主表
B
(C'(i), S'(i+1)) 其中 C(i) = C(D(i), S(i)) S(i+1) = S(D(i), S(i)) 来自替换表
2) D(i) = 88, ..., 255 且 S(i) = 1或4
(D(i), S(i))
A
(C(i), S(i+1)) 其中 C(i) = C(D(i), S(i)) S(i+1) = S(D(i), S(i)) (来自主表)
B
(C'(i), S'(i+1)) 其中 C'(i) = C(D(i), S'(i)) S'(i+1) = S(D(i), S'(i)) 且 S'(i) = 4 如果 S(i) = 1 1 如果 S(i) = 4 (来自主表)
(假定(C(i-1), C'(i))
满足RLL规则)
3) D(i) = 88, ..., 255 且S(i) = 2或3
(D(i), S(i))
A
(C(i), S(i+1)) 其中 C(i) = C(D(i), S(i)) S(i+1) = S(D(i), S(i)) (来自主表)

图 6

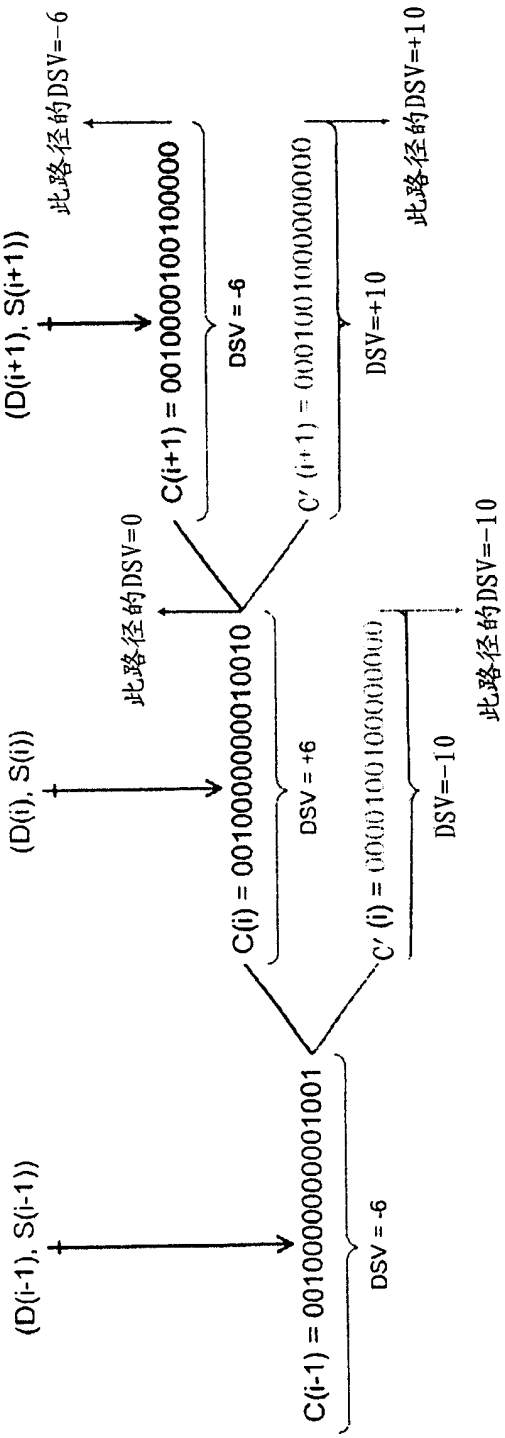

(D(i-1), S(i-1))
C(i-1) = 0010000000001001
DSV = -6
(D(i), S(i))
此路径的DSV=0
C(i) = 0010000000010010
DSV = +6
C′(i) = 0000100100000000
DSV=−10
此路径的DSV=−10
(D(i+1), S(i+1))
此路径的DSV=−6
C(i+1) = 0010000100100000
DSV = -6
C′(i+1) = 0010010000000000
DSV=+10
此路径的DSV=+10

图 7

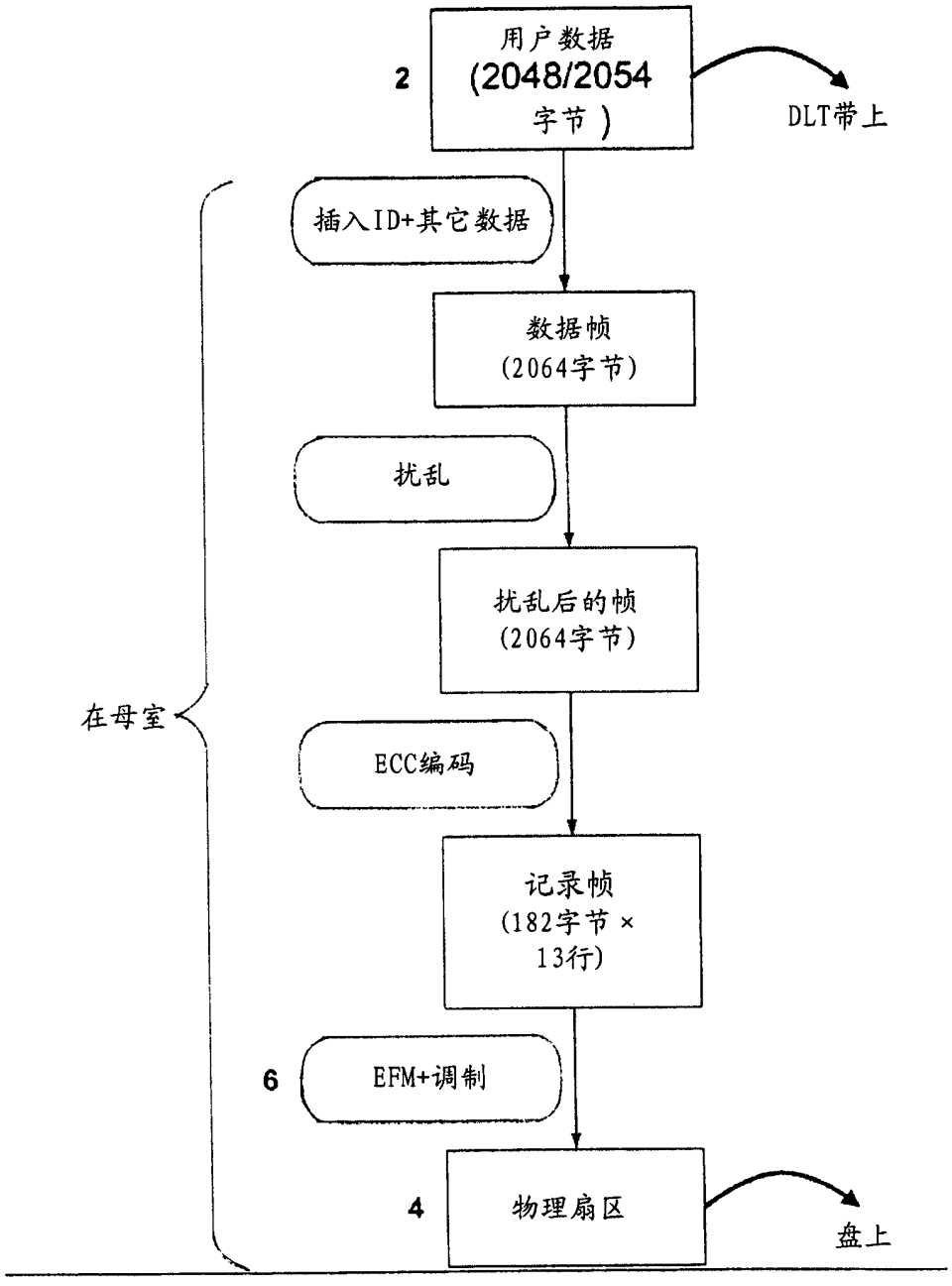

用户数据
(2048/2054
字节)
2
DLT带上
插入ID+其它数据
数据帧
(2064字节)
扰乱
扰乱后的帧
(2064字节)
在母室
ECC编码
记录帧
(182字节×
13行)
6
EFM+调制
4
物理扇区
盘上

图 8

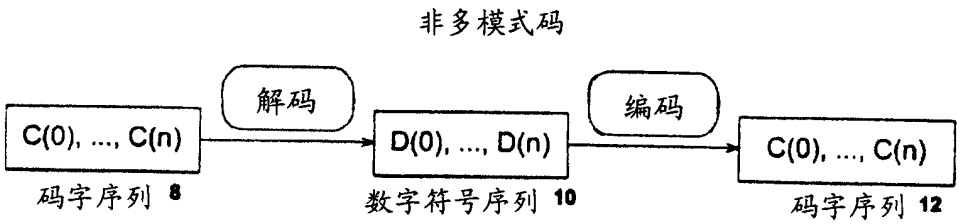

非多模式码
解码
编码
C(0), ..., C(n)
D(0), ..., D(n)
C(0), ..., C(n)
码字序列 8
数字符号序列 10
码字序列 12

图 9

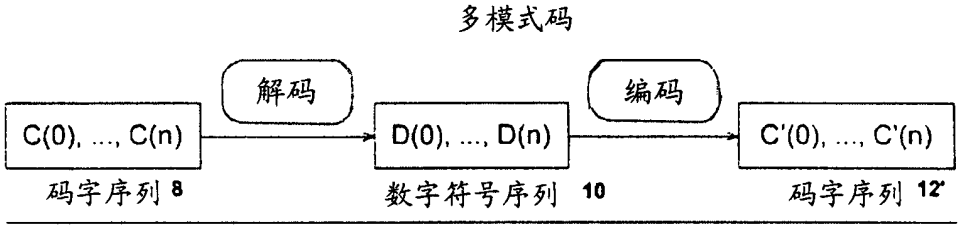

多模式码
解码
编码
C(0), ..., C(n)
D(0), ..., D(n)
C'(0), ..., C'(n)
码字序列 8
数字符号序列 10
码字序列 12'

图 10

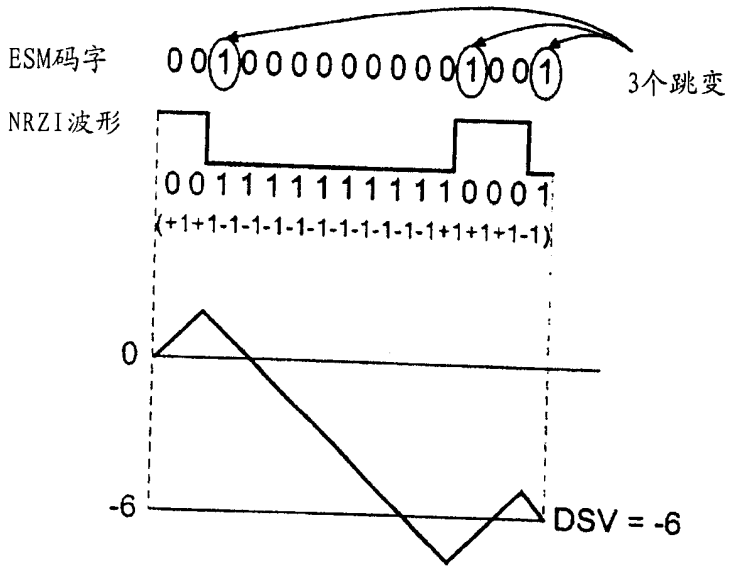
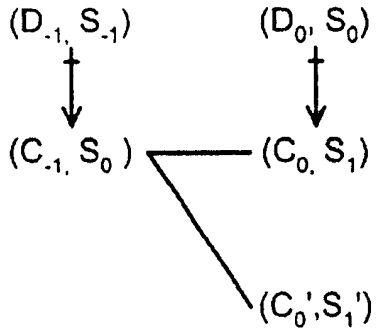
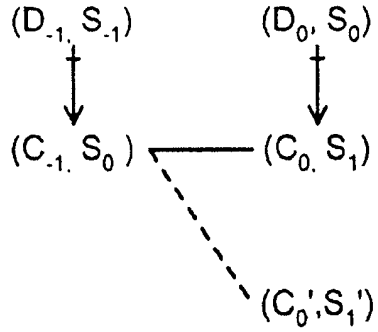

ESM码字
00100000000010001
3个跳变
NRZI波形
00111111111110001
(+1+1-1-1-1-1-1-1-1-1-1+1+1+1-1)
0
-6
DSV = -6

图 11

情况(1)



情况(2)



情况(3)

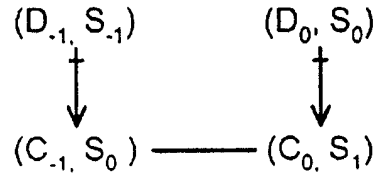
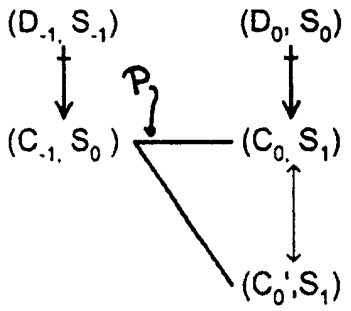
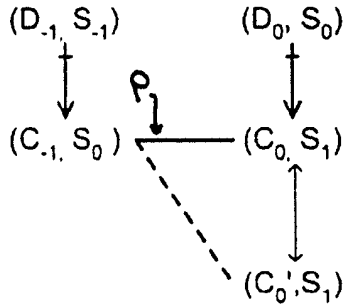


图 12

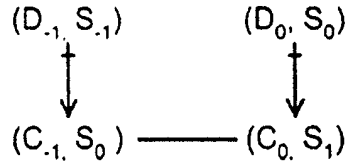
情况(1)



情况(2)



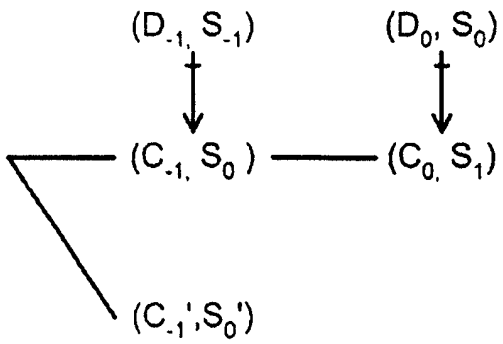
情况(3)



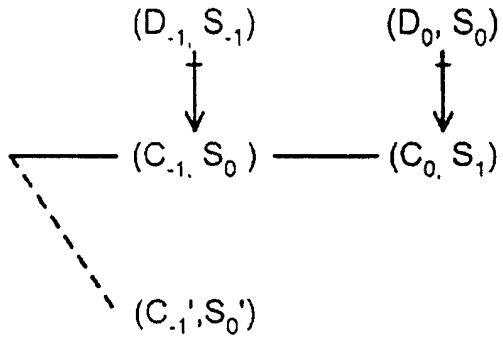
图例: ↕ 等价码字

图 13

情况(3.1)



情况(3.2)



情况(3.3)

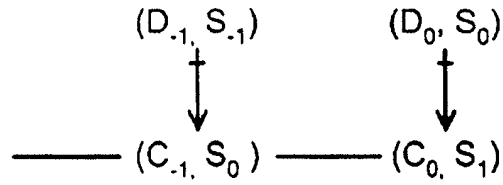
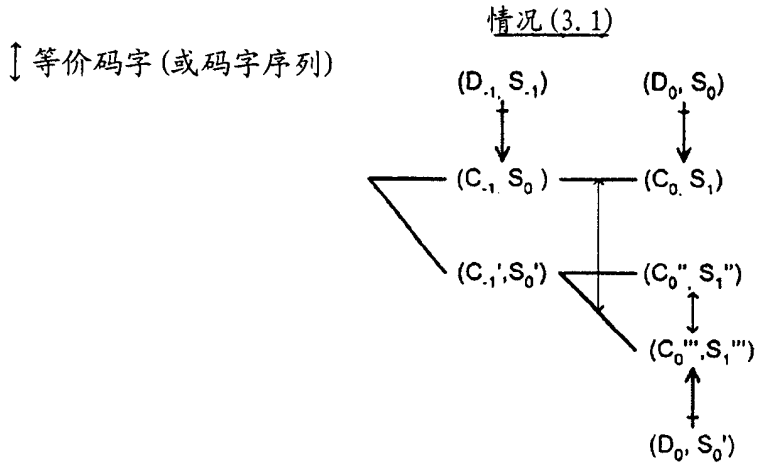
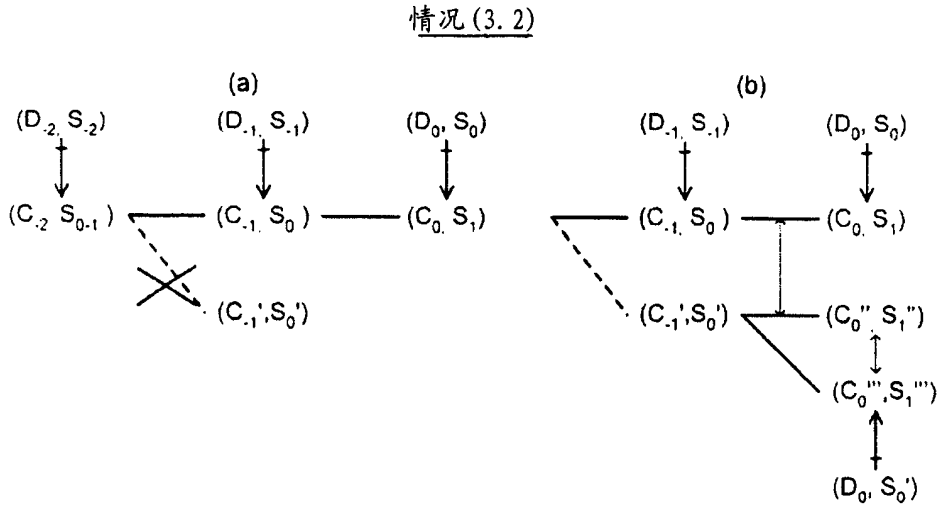


图 14


情况(3.1)
等价码字(或码字序列)
(D-1, S-1)
(D0, S0)
(C-1, S0)
(C0, S1)
(C-1', S0')
(C0", S1")
(C0"", S1"")
(D0, S0')


情况(3.2)
(a)
(b)
(D-2, S-2)
(D-1, S-1)
(D0, S0)
(C-2 S0-1)
(C-1, S0)
(C0, S1)
(C-1', S0')
(C0", S1")
(C0"", S1"")
(D0, S0')

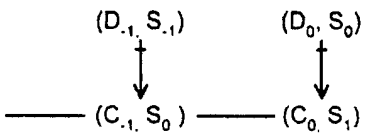
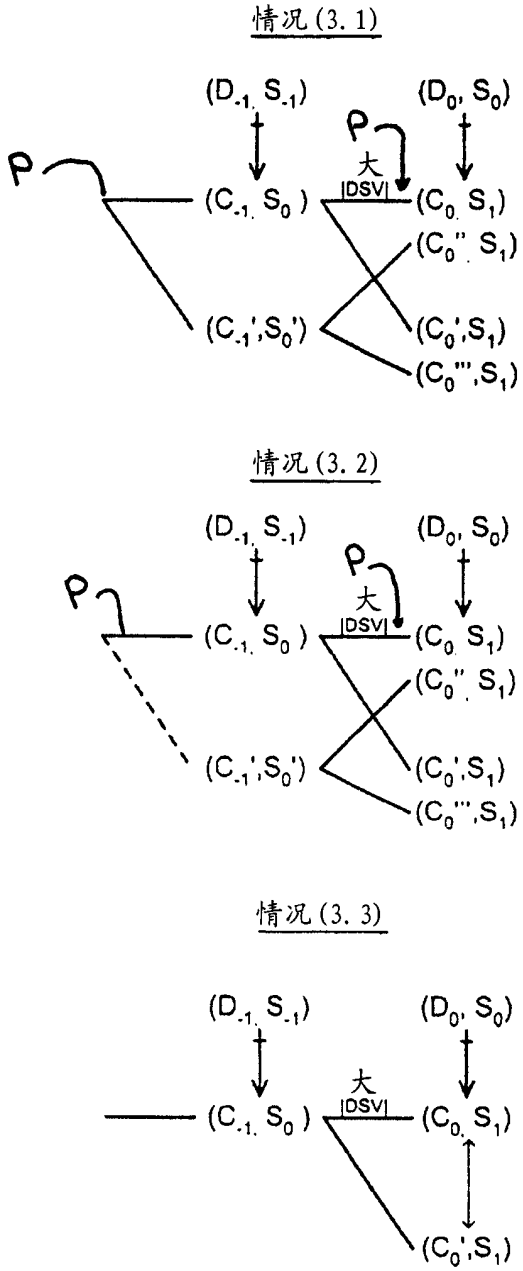

情况(3.3)
(D-1, S-1)
(D0, S0)
(C-1, S0)
(C0, S1)

图 15


情况(3.1)
(D-1, S-1)
(D0, S0)
P
大
|DSV|
(C-1, S0)
(C0, S1)
(C0'', S1)
(C-1', S0')
(C0', S1)
(C0''', S1)
情况(3.2)
(D-1, S-1)
(D0, S0)
P
大
|DSV|
(C-1, S0)
(C0, S1)
(C0'', S1)
(C-1', S0')
(C0', S1)
(C0''', S1)
情况(3.3)
(D-1, S-1)
(D0, S0)
大
|DSV|
(C-1, S0)
(C0, S1)
(C0', S1)

图例: ↕ 等价码字

图 16

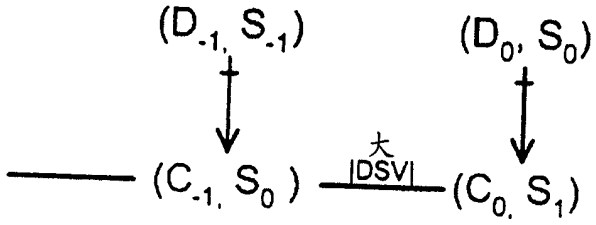


图 17

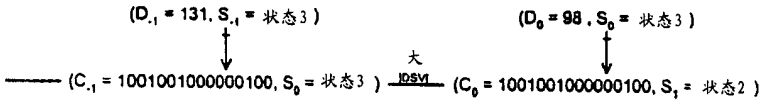

(D-1 = 131, S-1 = 状态3)
(D0 = 98 , S0 = 状态3)
(C-1 = 1001001000000100, S0 = 状态3)
大
(C0 = 1001001000000100, S1 = 状态2)

图 18

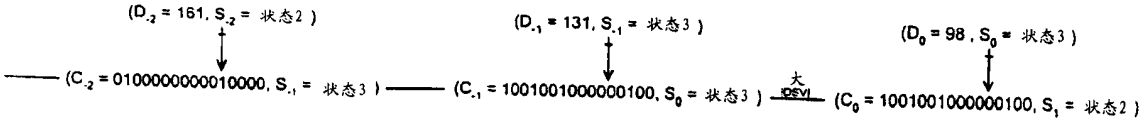

(D-2 = 161, S-2 = 状态2)
(D-1 = 131, S-1 = 状态3)
(D0 = 98 , S0 = 状态3)
(C-2 = 0100000000010000, S-1 = 状态3)
(C-1 = 1001001000000100, S0 = 状态3)
大
(C0 = 1001001000000100, S1 = 状态2)

图 19