

[19] 中华人民共和国国家知识产权局

[51] Int. Cl.

H03M 13/05 (2006.01)



[12] 发明专利说明书

专利号 ZL 200410057560.7

[45] 授权公告日 2009 年 7 月 1 日

[11] 授权公告号 CN 100508404C

[22] 申请日 2000.4.14

[21] 申请号 200410057560.7

分案原申请号 00800593.1

[30] 优先权

[32] 1999.4.16 [33] FI [31] 990863

[73] 专利权人 诺基亚西门子网络公司

地址 芬兰埃斯波

[72] 发明人 艾斯科·尼明宁 劳里·派尔蒂霍

[56] 参考文献

WO9909724A2 1999.2.25

US5790569A 1998.8.4

US4929946A 1990.5.29

CN1191684C 2005.3.2

US5889791A 1999.3.30

审查员 刘 力

[74] 专利代理机构 北京市金杜律师事务所

代理人 吴立明 罗世娜

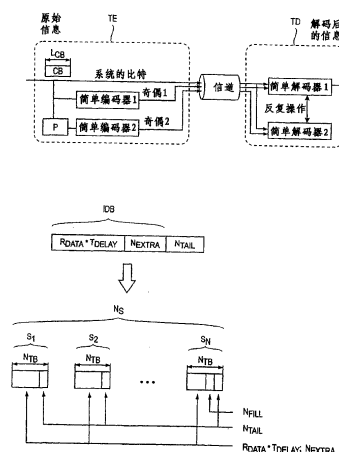
权利要求书 2 页 说明书 6 页 附图 1 页

[54] 发明名称

块编码器中的分段机制

[57] 摘要

一种利用块编码器 (TE) 将输入数据块 (IDB) 编码的方法。该块编码器能处理连续编码块 (CB)，编码块的长度有一个小于输入数据块 (IDB) 的长度的上限 (L_{CB})。该方法包括：(1) 在利用块编码器 (TE) 将输入数据块 (IDB) 的数据中的任何数据编码前，确定该输入数据块 (IDB) 的长度；(2) 将输入数据块 (IDB) 分成多段 ($S_1 \dots S_N$)，其中，所有的段其长度都基本相等，且所有的段都不大于上限 (L_{CB})；和 (3) 利用块编码器 (TE) 处理各个段 ($S_1 \dots S_N$)。如果最后一段 (S_N) 比其余段 ($S_1, S_2 \dots$) 短，则将一些填充比特 (N_{FILL}) 附加到最后一段 (S_N) 中，使得最后一段的长度等于其余段 ($S_1, S_2 \dots$) 的长度。



1. 一种利用块编码器 (TE) 将输入数据块 (IDB) 编码的方法, 所述块编码器能处理连续编码块 (CB), 编码块的长度有一个小于输入数据块 (IDB) 的长度的上限 (L_{CB});

其特征在于, 该方法包括以下步骤:

在利用所述块编码器 (TE) 将输入数据块 (IDB) 的数据中的任何数据编码前, 确定该输入数据块 (IDB) 的长度;

将输入数据块 (IDB) 分成多段 ($S_1...S_N$), 其中, 所有的段其长度都基本相等, 且所有的段都不大于所述上限 (L_{CB}); 和

利用所述块编码器 (TE) 处理各个段 ($S_1...S_N$).

2. 如权利要求1所述的方法, 其特征在于, 如果最后一段 (S_N) 比其余段 ($S_1, S_2...$) 短, 则将一些填充比特 (N_{FILL}) 附加到最后一段 (S_N) 中, 使得最后一段的长度等于其余段 ($S_1, S_2...$) 的长度。

3. 如权利要求1所述的方法, 其特征在于, 如果输入数据块 (IDB) 的长度不是所述上限 (L_{CB}) 的整倍数, 则将一些填充比特 (N_{FILL}) 附加到输入数据块 (IDB) 中, 使得其长度为所述上限 (L_{CB}) 的整倍数。

4. 如上述权利要求任一所述的方法, 其特征在于, 段数为 2^n , 其中 n 为正整数。

5. 一种分段装置, 用于将输入数据块 (IDB) 分段, 以便利用块编码器 (TE) 进行处理, 所述块编码器能处理连续编码块 (CB), 编码块的长度有一个小于输入数据块 (IDB) 的长度的上限 (L_{CB});

其特征在于, 该分段装置包括:

用于在将输入数据块 (IDB) 的数据中的任何数据输入到所述块编码器 (TE) 之前, 确定该输入数据块 (IDB) 的长度的装置;

用于将输入数据块 (IDB) 分成多个段 ($S_1...S_N$) 的装置, 其

中，所有的段其长度都基本相等，且所有的段都不大于所述上限 (L_{CB})；和

用于将各个段 ($S_1...S_N$) 输入到所述块编码器 (TE) 的装置。

块编码器中的分段机制

本申请是申请日为2000年4月14日，申请号为00800593.1，标题为“块编码器中的分段机制”的专利申请的分案申请。

技术领域

本发明涉及块编码器的方法和装置。

背景技术

众所周知，块编码器往往用于纠错。块编码器的一个例子是如参考文献1中所公开的所谓快速编码器。

图1是快速编码器TE的框图，该编码器通过（传输）信道与相应的快速解码器TD连接。典型的快速编码器将原始信息直接传送到信道。这些比特被称为系统比特。另外，快速编码器还利用简单编码器1和2加进冗余（奇偶），编码器2的前面加了一个交错器P，它置换原始信息的比特。不过，块编码器的细节对理解本发明而言无关紧要，有关情况可参照相关文献。

与流式编码器不同，块编码器可同时处理一个或多个数据块。长度超过块编码器的块长度的输入数据块必须划分为更小的段，使得所有的段都不大于块编码器的块长度。因此，块编码器特别适合于具有固定输入块长度的应用。使用块编码器的问题在于，它们不适宜于具有可变（动态）输入块长度的应用。换言之，如何处理输入数据块的最后几段仍是个未解决的问题。

发明内容

本发明的目的在于，提供一种使块编码器适用于具有可变的（动态的）输入块长度的应用的机制。该机制将是通用的，以便能适应各种块编码器。

为实现本发明的目的，提供了一种利用块编码器将输入数据块

编码的方法，所述块编码器能处理连续编码块，编码块的长度有一个小于输入数据块的长度的上限；该方法包括以下步骤：在利用所述块编码器将输入数据块的数据中的任何数据编码前，确定该输入数据块的长度；将输入数据块分成多段，其中，所有的段其长度都基本相等，且所有的段都不大于所述上限；和利用所述块编码器处理各个段。

在一个优选实施例中，如果最后一段比其余段短，则将一些填充比特附加到最后一段中，使得最后一段的长度等于其余段的长度。

在一个优选实施例中，如果输入数据块的长度不是所述上限的整倍数，则将一些填充比特附加到输入数据块中，使得其长度为所述上限的整倍数。

在一个优选实施例中，段数为 2^n ，其中 n 为正整数。

还提供了一种分段装置，用于将输入数据块分段，以便利用块编码器进行处理，所述块编码器能处理连续编码块，编码块的长度有一个小于输入数据块的长度的上限；该分段装置包括：用于在将输入数据块的数据中的任何数据输入到所述块编码器之前，确定该输入数据块的长度的装置；用于将输入数据块分成多个段的装置，其中，所有的段其长度都基本相等，且所有的段都不大于所述上限；和用于将各个段输入到所述块编码器的装置。

一种简单的方案是对输入数据块的段进行填充以达到块编码器的块长度。假定，块长度为8千比特（kb），那么14kb输入数据块可分为8kb的第一段和由6kb的净长度及2kb的填充（填补）比特组成的第二（最末）段。这一简单的方案的好处在于，块编码器不必去适应变化的输入块长度。

本发明基于这样的思想：对于长度超过块编码器的块长度的输入数据块：

- 1) 在开始编码前，确定输入数据块的长度；和
- 2) 将输入数据块分成一些长度大致相等的段，使得所有的段都

不大于块编码器的块长度。

根据本发明的一种优选实施方式，将输入数据块分为尽可能少的段数。换言之，这些段应尽可能大。

根据本发明的另一实施方式，将输入数据块分为 2^n 个段，其中 n 为正整数。

根据本发明的又一优选实施方式，如果划分输入数据块时产生的最后一段比其余段短，则用一些填充比特来填补该输入数据块或最后一段，直到最后一段的长度等于其余段的长度。不过，与简单方案相比，该最后一段并不是被填补到块编码器的满块长度（除非其余段碰巧也是这个长度）。

附图说明

下面，结合一些优选实施方式并参照附图来详述本发明，其中：

图1是快速编码器的框图；和

图2示出了将输入数据块分为适用于快速编码器的一些段。

具体实施方式

下面，针对一种快速编码器来描述本发明的各种实施方式，这种快速编码器的一个例子如参考文献1中所述。不过，快速编码器或任何其他编码器的细节对理解本发明而言无关紧要。

通过限定编码块的长度，可以简化快速编码器（和相应的解码器）的实现。编码块的长度其适当的值为8192比特，其中包括用户数据、可能的检错字段（CRC）和结尾。以下是所用的命名协定：

N_{TAIL} = 结尾比特数

T_{DELAY} （秒） = 用户数据块的长度

R_{DATA} （比特/秒） = 业务的用户数据速率

N_{EXTRA} = 附加到原始用户数据中的其他比特（CRC等）数

L_{CB} = 编码块的最大长度。

下列条件必须满足:

$$R_{DATA} * T_{DELAY} + N_{EXTRA} + N_{TAIL} \leq L_{CB} \quad [1]$$

如果该条件不能满足,那么,所要编码的数据必须分段,使得各段都满足该条件。段数 N_S 必须满足下列条件:

$$\text{上舍入}((R_{DATA} * T_{DELAY} + N_{EXTRA})/N_S) + N_{TAIL} \leq L_{CB} \quad [2]$$

最好选择满足不等式[2]的最小 N_S 。可根据下式计算 N_S :

$$N_S = \text{上舍入}((R_{DATA} * T_{DELAY} + N_{EXTRA})/(L_{CB} - N_{TAIL}))[3]$$

可能碰巧所有编码块没有以相同的长度结束,即 $(R_{DATA} * T_{DELAY} + N_{EXTRA})/N_S$ 不是一个整数。在这种情况下,至少有两种可行的解决方案,称之为算法A和B。在算法A中,允许最后一段有与其他段不同的长度。在算法B中,将 N_{FILL} 个填充比特(如“0”)附加到输入数据中,使得, $(R_{DATA} * T_{DELAY} + N_{EXTRA} + N_{FILL})/N_S$ 为尽可能小的整数。(或者说,可将填充比特附加到分段后的最后一段中)

算法A:

算法A允许最后一段比其他段短。算法A采用下列输入量:

R_{DATA} = 用户数据速率(比特/秒)

T_{DELAY} = 编码用户数据块的长度(秒)

N_{EXTRA} = 编码前附加到用户数据中的附加数据(比特)

N_{TAIL} = 附加到编码块中的尾比特数

算法A得到下列输出量:

N_S = 段数

N_{TB} = 除最后一个块外的快速编码器输入块中的比特数

$N_{LAST-TB}$ = 最后一个快速编码器输入块中的比特数

在算法A中,将进行以下计算:

设 $N_S = \text{上舍入}((R_{DATA} * T_{DELAY} + N_{EXTRA})/(L_{CB} - N_{TAIL}))$

设 $N_{TB} = \text{上舍入}((R_{DATA} * T_{DELAY} + N_{EXTRA})/N_S) + N_{TAIL}$;

设 $N_{REM} = ((R_{DATA} * T_{DELAY} + N_{EXTRA})/N_S)$ 的余数;

如果 $N_{REM} \neq 0$, 则 $N_{LAST-TB} = N_{TB} - N_S + N_{REM}$, 否则 $N_{LAST-TB} =$

N_{TB}

结束。

如果采用算法A，那么，由于输入到快速编码器的最后输入段比其他输入段短，因此需要一种自适应快速交错器。编码器的输出中的系统的比特数为：

$$R_{DATA} * T_{DELAY} + N_{EXTRA} + N_S * N_{TAIL} [4]$$

因此，除了每段的结尾的比特之外没有附加比特。

算法B：

如图2中所示，在算法B中，输入到快速编码器的所有输入段长度都相等。算法B的输入量为：

R_{DATA} = 用户数据速率(比特/秒)

T_{DELAY} = 编码用户数据块的长度(秒)

N_{EXTRA} = 编码前附加到用户数据中的附加数据(比特)

N_{TAIL} = 附加到编码块中的尾比特数

算法B的输出量为：

N_S = 段数

N_{TB} = 快速编码器输入块中的比特数

N_{FILL} = 最后一个快速编码器输入块中的填充比特(如“0”)

数

在算法B中，将进行以下计算：

设 N_S = 上舍入 $((R_{DATA} * T_{DELAY} + N_{EXTRA}) / (LCB - N_{TAIL}))$

设 N_{TB} = 上舍入 $((R_{DATA} * T_{DELAY} + N_{EXTRA}) / N_S) + N_{TAIL}$

设 N_{REM} = $((R_{DATA} * T_{DELAY} + N_{EXTRA}) / N_S)$ 的余数

如果 $N_{REM} \neq 0$ ，则将 $N_{FILL} = (N_S - N_{REM})$ 个“0”比特插入到输入数据的末端，否则 $N_{FILL} = 0$ 。

输入到快速编码器的所有输入段长度都相等，因此对所有的段而言都可以利用相同的快速交错器。这样，在快速编码器的输出端，整个信道交换块上的系统的比特数为：

$$N_S * (\text{上舍入}((R_{DATA} * T_{DELAY} + N_{EXTRA}) / N_S) + N_{TAIL}) [5]$$

因此，除了结尾比特之外还有一些附加比特。

算法A和B的修改：

在上述算法A和B中，通过选择尽可能少的段数 N_S ，使输入到快速编码器的输入段的长度最大。在某些情况下，可以优先采用为2的次方的段数 N_S ，但这会使输入到快速编码器的输入段变短。在这种情况下，可用以下三步来取代上述算法A和B中的第一步：

设 $n_S = \text{上舍入}((R_{\text{DATA}} * T_{\text{DELAY}} + N_{\text{EXTRA}})/(L_{\text{CB}} - N_{\text{TAIL}}))$ ；

设 $m = \text{上舍入}(\log_2 n_S)$ ；

设 $N_S = 2^m$ 。

参考文献：

1. C.Berrou, A.Glavieux, P.Thitimajshima: Near Shannon limit error-correcting coding and decoding : Turbo-codes, IEEE International Conference on Communications, ICC 1993, Geneva, Switzerland 23-26 May, 1993, Vol.2, pp.1064-1070.

所有参考文献在此作为参考。

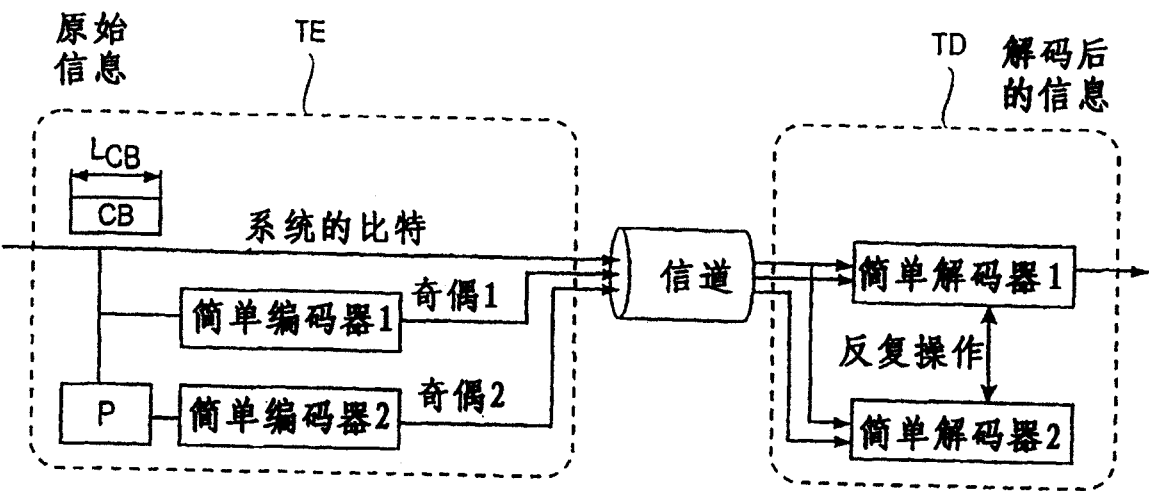


图1

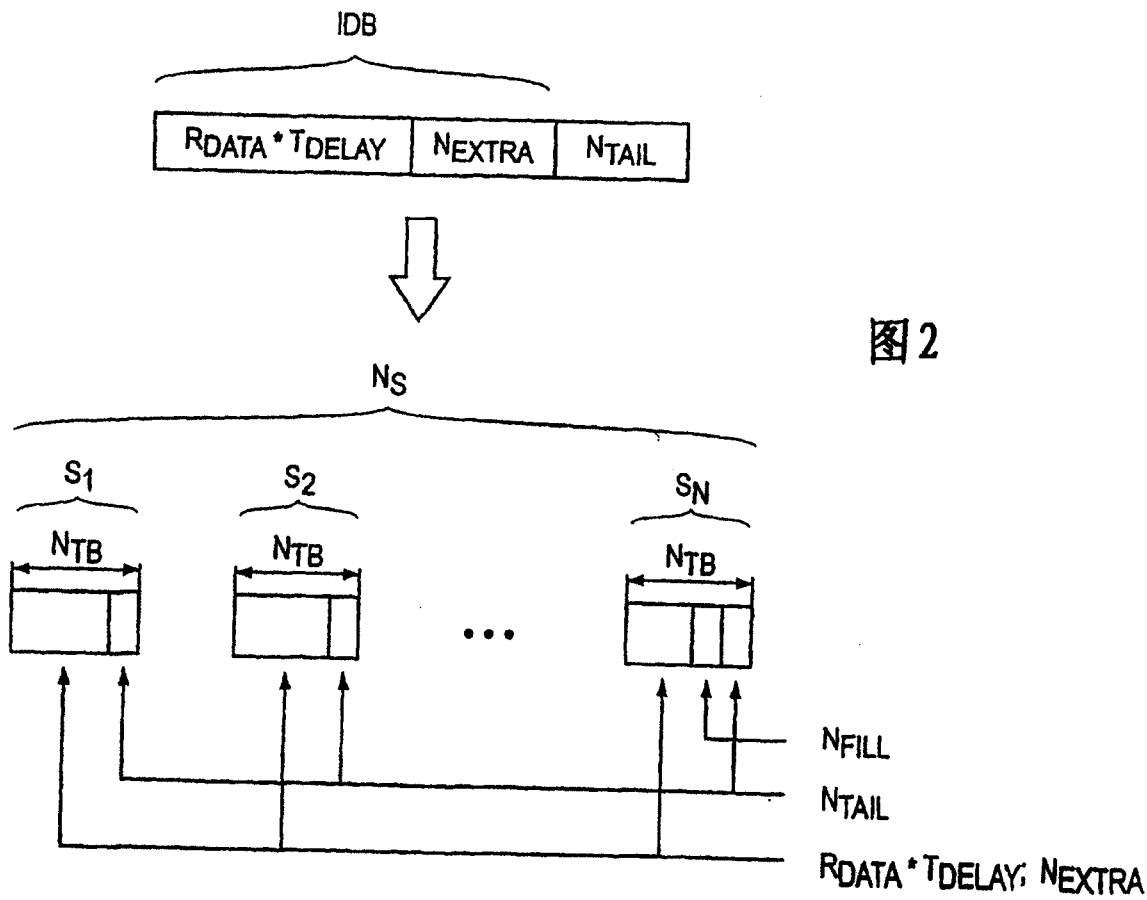


图2