

[19] 中华人民共和国国家知识产权局



[12] 发明专利申请公布说明书

[21] 申请号 200580033473.0

[43] 公开日 2007 年 9 月 26 日

[51] Int. Cl.
H03M 13/00 (2006.01)
H03M 13/03 (2006.01)

[11] 公开号 CN 101044688A

[22] 申请日 2005.8.1

[21] 申请号 200580033473.0

[30] 优先权

[32] 2004.8.2 [33] US [31] 10/909,753

[86] 国际申请 PCT/US2005/027526 2005.8.1

[87] 国际公布 WO2006/017555 英 2006.2.16

[85] 进入国家阶段日期 2007.4.2

[71] 申请人 高通弗拉里奥恩技术公司

地址 美国加利福尼亚州

[72] 发明人 汤姆·理查森

瓦拉迪莫·诺维齐科夫

[74] 专利代理机构 永新专利商标代理有限公司
代理人 王英

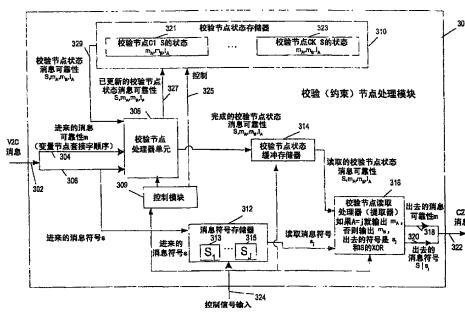
权利要求书 7 页 说明书 25 页 附图 8 页

[54] 发明名称

节省存储器的 LDPC 译码方法和装置

[57] 摘要

描述了用于实现节省存储器的 LDPC 译码的方法和装置。根据本发明，将消息信息储存在压缩状态(310)，用于校验节点处理操作。全部更新校验节点(321)的状态，然后对校验节点(321)的状态进行提取处理(316)，以产生校验节点到变量节点消息。从变量节点收到的消息的符号可以由本发明的校验节点处理器模块(312)储存，用于消息提取。校验节点处理器(308)可以按照变量节点顺序(304)处理消息，从而允许变量节点处理器和校验节点处理器按照同一顺序对消息进行操作，减小或消除对校验节点和变量节点之间传递的消息进行缓存和/或重新排序的必要。还描述了图结构，这些图结构允许对一个图迭代的校验节点处理在前一个图迭代完成之前进行。



1. 一种用于进行低密度奇偶校验（LDPC）译码操作的装置，包括：

校验节点处理器模块，该模块包括：

- i) 校验节点状态存储器，包括用于多个校验节点的多个消息状态存储器存储单元，每个校验节点状态存储单元对应于单独一个校验节点并且包括用于储存的第一和第二位置，该第一和第二位置用于储存第一和第二消息幅度值，该第一和第二消息幅度值对应于给所述校验节点的消息，其中所述校验节点状态存储器对应于该校验节点，每个节点状态存储单元还包括符号存储器位置，该符号存储器位置用于储存积累的符号值，该符号值对应于所述校验节点状态存储单元所对应的校验节点；
- ii) 校验节点处理器单元，用于基于收到的变量节点到校验节点状态消息的内容，更新所述校验节点状态存储器中储存的状态；以及
- iii) 控制模块，连接到所述校验节点状态存储器，用于控制该校验节点状态存储器，以输出对应于一个校验节点的校验节点状态，该校验节点和要处理的变量到校验节点消息是同一个节点，所述校验节点状态是从所述校验节点状态存储单元中的一个单元输出的，该单元对应于与要处理的所述变量到校验节点消息相同的节点。

2. 如权利要求 1 所述的装置，还包括：

消息符号存储器，用于储存对应于校验节点的每个收到的消息中包括的符号位，用于产生校验节点到变量节点消息。

3. 如权利要求 2 所述的装置，还包括：

校验节点状态缓冲存储器，用于储存和至少一个校验节点相对应的已全部更新的校验节点状态信息。

4. 如权利要求 3 所述的装置，其中和所述至少一个校验节点相

对应的已全部更新的校验节点状态信息是根据 M 个变量节点到校验节点消息中的每一个产生的状态信息，其中 M 是和所述至少一个校验节点相对应的边的数量。

5. 如权利要求 3 所述的装置，还包括：

校验节点到变量节点消息提取器，连接到所述校验节点状态缓冲存储器和所述消息符号存储器，用于根据对应于所述至少一个校验节点的所述已全部更新的校验节点状态信息，以及根据对应于所述至少一个校验节点的储存的符号消息信息，产生校验节点到变量节点消息，针对对应于所述至少一个校验节点而接收到的每一个校验节点到变量节点消息，产生一个校验节点到变量节点消息。

6. 如权利要求 5 所述的装置，其中所述控制模块控制所述校验节点状态存储器，将对应于校验节点的已更新状态储存到从中读取了对应于该校验节点的状态信息的同一个消息状态存储器存储单元中。

7. 如权利要求 1 所述的装置，其中对于用于控制译码的 LDPC 图结构中包括的每个校验节点，所述校验节点状态存储器包括一个校验节点状态存储器存储单元。

8. 如权利要求 1 所述的装置，其中对于用于控制译码的 LDPC 图结构中包括的每个校验节点，所述校验节点状态存储器包括一个校验节点状态存储器存储单元。

9. 如权利要求 2 所述的装置，其中对于用于控制译码的 LDPC 图结构中包括的每个消息边，所述消息符号存储器包括一个符号位存储位置。

10. 如权利要求 9 所述的装置，还包括：

变量节点处理器，连接到所述校验节点处理模块，所述变量节点

处理器模块从该校验节点处理模块接收校验节点到变量节点消息，并且产生提供给该校验节点处理模块的变量节点到校验节点消息。

11. 如权利要求 10 所述的装置，还包括：

译码器控制模块，连接到所述校验节点处理模块，并且连接到所述变量节点处理器模块，该译码器控制模块控制所述校验节点处理模块和所述变量节点处理器模块，以便按照变量节点顺序处理消息。

12. 如权利要求 11 所述的装置，其中所述变量节点处理器包括并行排列的多个变量节点处理单元。

13. 如权利要求 12 所述的装置，其中所述校验节点处理器模块包括并行排列的多个校验节点处理单元。

14. 如权利要求 13 所述的装置，其中在所述变量节点处理器模块和所述校验节点处理器模块之间排列有至少一个消息重新排序单元，用于对所述变量节点处理器和所述校验节点处理器之间并行传递的消息进行重新排序。

15. 如权利要求 14 所述的装置，其中所述译码器控制模块包括控制逻辑，用于根据储存的消息重新排序信息，控制所述消息重新排序单元。

16. 一种进行低密度奇偶校验译码操作的方法，包括以下步骤：

在存储器中储存和多个校验节点收到的消息相对应的消息状态信息；

接收到所述多个校验节点之一的校验节点输入消息；

基于要将所述收到的消息给到的校验节点，选择用于状态更新操作的所述多组状态信息之一；

从所述存储器提取所选择的那组消息状态信息；以及

根据所述收到的消息更新所选择的那组消息状态信息。

17. 如权利要求 16 所述的方法，还包括：

将所述已更新的那组消息状态信息写入从中提取过所述消息状态信息的存储器位置。

18. 如权利要求 17 所述的方法，还包括：

按照第一序列，对收到的多则消息中的每一则，顺序重复所述接收、选择、提取和更新步骤，该第一序列对应于和所述收到的消息对应的边连接到代表 LDPC 码的图中的变量处理节点上去的顺序。

19. 如权利要求 18 所述的方法，其中和对应于所述收到的消息的边连接到代表所述 LDPC 码的图中的变量处理节点上去的顺序相对应的所述序列不同于第二序列，其中对应于所述收到的消息的边按照该第二序列连接到代表所述 LDPC 码的图中的约束处理节点上去。

20. 如权利要求 18 所述的方法，还包括：

对和校验节点相对应的至少一组状态信息进行校验节点到变量节点消息提取操作，已经为该校验节点收到了完整的一组变量到校验节点消息。

21. 如权利要求 20 所述的方法，还包括：

进行所述提取操作，产生多个校验节点到变量节点消息，连续产生给所述多个变量节点中的单独一个的校验节点到变量节点消息，以产生给所述多个变量节点中的所述单独一个的校验节点到变量节点消息序列。

22. 如权利要求 16 所述的方法，还包括：

在完成至少一个其它校验节点的状态更新之前，全部完成第一校验节点的状态更新，以产生对应于该第一校验节点的一组已全部更新

的状态，当针对和一个校验节点相对应的多个消息边中的每一个将该校验节点的状态更新了一次的时候，该校验节点的状态更新全部完成。

23. 如权利要求 16 所述的方法，还包括：

在所述至少一个其它校验节点的所述状态更新全部完成之前，更新和所述第一校验节点相对应的另一组状态，作为变量到校验节点消息处理的第二次迭代的一部分。

24. 如权利要求 23 所述的方法，还包括以下步骤：

缓存和所述第一校验节点相对应的所述已全部更新的状态；以及根据所述缓存的已全部更新的状态提取校验节点到变量节点消息。

25. 如权利要求 24 所述的方法，其中根据所述缓存的已全部更新的状态提取校验节点到变量节点消息的所述步骤包括：

根据所述已全部更新的状态以及和用于产生所述已全部更新的状态的多个变量节点到校验节点消息相对应的储存的符号信息，产生多个出去的消息。

26. 如权利要求 16 所述的方法，还包括：

在完成和第二组校验节点相对应的状态更新之前，全部完成第一组校验节点的状态更新，当针对和一个校验节点相对应的多个消息边中的每一个将该校验节点的状态更新了一次的时候，该校验节点的状态更新全部完成。

27. 如权利要求 26 所述的方法，其中所述第一组和第二组校验节点中的每一组都包括代表所实现的用于控制译码的 LDPC 码结构的 LDPC 图中校验节点总数的至少 20%。

28. 如权利要求 16 所述的方法，其中所述更新操作是作为和第一组校验节点相对应的部分更新状态信息进行的。

29. 如权利要求 28 所述的方法，其中第一组校验节点的所述更新是利用第一组变量到校验节点消息在第一时间周期中进行的，该方法还包括：

在第二时间周期中更新和第二组校验节点相对应的状态信息，所述第二组校验节点只包括所述第一组校验节点中不包括的校验节点，所述第二时间周期跟随在所述第一时间周期之后。

30. 如权利要求 29 所述的方法，还包括：

在所述第二时间周期中，根据和所述第一组校验节点相对应的已更新状态信息提取校验节点到变量节点消息。

31. 如权利要求 30 所述的方法，其中所述第一和第二时间周期在长度上相等。

32. 如权利要求 30 所述的方法，其中所述第一和第二时间周期被第三时间周期分开，在该第三时间周期中更新和第三组校验节点对应的状态。

33. 如权利要求 30 所述的方法，其中所述第一和第二组校验节点中的每一组都包括和用于控制译码的 LDPC 码相对应的图中校验节点的至少 10%。

34. 如权利要求 30 所述的方法，其中所述第一和第二组校验节点中的每一组都包括和用于控制译码的 LDPC 码相对应的图中校验节点的至少 20%。

35. 如权利要求 30 所述的方法，其中所述第一时间周期少于处

理 N 个变量到校验节点消息所需要的时间的 40%，其中 N 等于和用于控制译码的 LDPC 码相对应的图中消息边的总数。

36. 如权利要求 30 所述的方法，其中所述第一时间周期少于处理 N 个变量到校验节点消息所需要的时间的 20%，其中 N 等于和用于控制译码的 LDPC 码相对应的图中消息边的总数。

节省存储器的 LDPC 译码方法和装置

背景技术

几乎所有形式的电子通信和存储系统都要使用纠错码。通过将冗余性引入数据流，纠错码能够补偿这些系统中信息传递的固有不可靠性。香农为纠错建立了数学基础。香农发展了信道的数学概念，用随机过程来模拟在这些信道中通信系统信号的失真。香农的最基本的结果是噪声信道定理），该定理为信道定义了容量，这个容量说明通过该信道能够可靠地传送信息的最大速率。以接近容量的速率进行可靠的传输需要使用纠错码。因此，设计了纠错码来实现足够的可靠性，同时尽可能地接近容量。实现纠错码的复杂性在于，在纠错码的实际应用中另一个因素开始起作用。来源于 Turbo 码这一发明的纠错编码系统的最新进展，以及低密度奇偶校验（LDPC）码的随后重新发现和开发，为编码系统提供了能够相当接近香农容量的能够实现的复杂性。

LDPC 码可以用二分图来很好地表示，二分图常常叫做 Tanner 图，比如图 1 所示的图 100。在 Tanner 图中，一组节点，即变量节点 102，对应于码字的比特；而另一组节点，即约束节点 106，有时叫做校验节点，则对应于定义这个码的这组奇偶校验约束。图中的边 104 把变量节点连接到约束节点。如果变量节点和约束节点由图中的一个边连接，就说它们是邻居。通常假设一对节点最多由一个边连接。LDPC 码可以用奇偶校验矩阵 202 等效地表示。图 2 给出了奇偶校验矩阵表示的一个实例，在并且只有在 $Hx = 0$ 的时候，其中标出的矢量 x 208 是一个码字。

每个变量节点与码字的一个比特相联系。在一些情况下，这些比特中的一些有可能被收缩（punctured）。收缩的比特在特定的码结构中可能是需要的，将它们从发射的码字中排除。

对于每个约束节点，在并且只有在和约束相邻（通过它们与变量节点的联系）的那些比特加起来等于 0 模 2，也就是说它们包括偶数个 1 的情况下，和变量节点序列一一对应地联系的比特序列才是这个码的一个码字。

用于对 LDPC 码字进行译码的译码器和译码算法通过在图内沿着那些边交换消息，并且通过基于进来的消息在那些节点处进行计算来更新这些消息。将把这些算法笼统地叫做消息传递算法。一开始给图中的每个变量节点提供一个软比特，叫做接收值，它表明通过从例如通信信道的观察确定的，相联系的比特的值的估计。理想情况下，分开的比特的估计在统计意义上是独立的。在实际中，会违背这一思想，并且会经常违背这一思想。接收值的集合构成接收字。为了这一申请的目的，我们用接收字来标识通过例如通信系统中的接收机观察到的信号。

到节点即变量节点或约束节点上的边数叫做这个节点的度。规则图或码的所有变量节点都具有相同的度（例如 j ），并且所有约束节点具有相同的度（例如 k ）。在这种情况下，我们说这个码是一个 (j, k) 规则码。这些码是 Gallager (1961) 一开始考虑的那些码。和“规则”码相对，不规则码是具有不同度的约束节点和/或变量节点。例如，一些变量节点可以具有度 4，其它的具有度 3，另一些具有度 2。

尽管不规则码表示和/或实现起来可能更加复杂，但是和规则 LDPC 码相比较，已经证明不规则 LDPC 码能够提供卓越的纠错/检错性能。

显然，LDPC 编码产生的接收字可以通过对它们进行 LDPC 译码操作，例如纠错和检错，来进行处理，产生原始码字的重构版本。然后可以对重构的码字进行数据译码处理，恢复最初编码的原始数据。数据译码过程可以是例如仅仅是从重构的码字中选择那些比特的一个具体子集。

如上所述，LDPC 译码操作一般都包括消息传递算法。有许多可能有用的消息传递算法，这些算法的使用不限于 LDPC 译码。

为了帮助理解在后面讨论的本发明，下面我们将针对置信传播进

行简短的数学描述。

可以按照如下方式表述（二进制）LDPC 码的置信传播。把沿着图的边传输的消息解释为和这个变量节点相联系的比特的对数似然 $\log \frac{p_0}{p_1}$ 。在这里， (p_0, p_1) 表示相联系的比特上的条件概率分布，其中 p_x 表示这个比特取值 x 的概率。接收机提供给译码器的软比特也是以对数似然的形式给出的。这样，接收值，也就是接收字的单元，是观察到通信信道提供的比特的条件下，相联系的比特的对数似然。一般而言，消息 m 表示对数似然 m ，接收值 y 表示对数似然 y 。对于收缩的比特，将接收值 y 的对数似然设置成 0，表明 $p_0 = p_1 = 1/2$ 。

让我们考虑置信传播的消息传递规则。对于从校验节点去往变量节点的消息，将消息表示为 m^{C2V} ，从变量节点去往校验节点的消息表示为 m^{V2C} 。考虑具有 d 个边的变量节点。对于每个边 $j = 1, \dots, d$ ，令 $m^{C2V}(j)$ 表示边 j 上进来的消息。在译码过程的初始化中，对于每个边我们设定 $m^{C2V} = 0$ 。总之，从变量节点出去的消息由下式给出：

$$m^{V2C}(j) = y + (\sum_{i=1}^d m^{C2V}(i)) - m^{C2V}(j).$$

对应于这一操作，来自一个节点的出去的已译码软值（不是边消息）由 $x_{out} = y + (\sum_{i=1}^d m^{C2V}(i))$ 给出。与这一输出相联系的出去的硬判决是从 x_{out} 的符号获得的。

在校验节点处，用消息的“符号”和幅度来表示它们常常更加方便。于是，对于消息 m ，令 $m_p \in GF[2]$ 表示消息的“奇偶性”，也就是说，如果 $m \geq 0$ ，那么 $m_p = 0$ ，并且如果 $m < 0$ ，那么 $m_p = 1$ 。另外，用 $m_r \in [0, \infty]$ 表示 m 的幅度。这样一来，我们有 $m = -1^{m_p} m_r$ 。在校验

节点处， m_p 和 m_r 的更新都是分别进行的。对于度 d 校验节点，我们有：

$$m_p^{C2V}(j) = (\sum_{i=1}^d m_p^{V2C}(i)) - m_p^{V2C}(j),$$

其中所有相加都是在 GF[2] 上进行的，并且

$$m_r^{C2V}(j) = F^{-1}((\sum_{i=1}^d F(m_r^{V2C}(i))) - F(m_r^{V2C}(j))),$$

其中相加是实的，并且我们定义： $F(x) := \ln \coth(x/2)$ 。要注意， F 是它自己的逆，也就是说： $F^{-1}(x) = F(x)$ 。

在 LDPC 领域里常常提到的一个算法是所谓的 MIN-SUM 算法。在这个算法中，在校验节点处的更新操作可以用数学表达式表示为：

$$m_r^{C2V}(j) = \min \{ (\bigcup_{i=1}^d m_r^{V2C}(i)) \setminus m_r^{V2C}(j) \}.$$

这样，C2V 消息的可靠性等于沿着其它边进来的 V2C 消息的最小可靠性。为了实现这一算法，在校验节点处保存最小和第二小可靠性（如果对于至少两个进来的消息都出现了这个值，那么它们可以是同样的值）以及提供具有最小可靠性的进来的消息的边的标识就足够了。最不可靠的进来的消息进来的边上出去的 C2V 消息的可靠性，等于第二小的进来的可靠性，并且所有其它边上出去的 C2V 消息的可靠性等于最小可靠性。

在 6,633,856 号美国专利中描述了一种 LDPC 译码器体系结构。在这种体系结构中，从校验节点向变量节点发送的消息储存在存储器中。如果这些消息包括例如 5 个比特，并且校验节点具有度 K ，那么用于这些消息的存储器有 $5K$ 个比特。

从实现和成本的角度看，一般都需要按照构造起来简单，需要较少量的硬件这种方式来实现 LDPC 译码器。在许多译码器设计中，存储器是一个主要组件。一般都希望实现译码器所需要的存储器最少，

以便降低硬件成本。

尽管减少存储器很重要，但是在减少所使用的存储器的同时，常常需要避免使得处理延迟变得无法接受，这种延迟会导致无法满足一个或多个真实世界译码时间约束。

基于以上讨论，很显然希望有节省存储器的方法和装置，能够用它们来简化 LDPC 译码器的实现，和/或得到能够以较少的存储器进行译码操作的 LDPC 译码器。另外，还希望有方法来避免给使用节省存储器的译码器的 LDPC 译码过程带来过分的延迟。

发明内容

本发明针对的是以节省存储器的方式进行 LDPC 译码操作的方法和装置。本发明的各个特征都是针对以节省存储器的方式实现的校验节点处理方法和装置，例如，采用一种或多种消息信息压缩和/或解压缩技术。本发明的其它特征是针对通过采用具有避免将明显的延迟引入译码处理的码结构的 LDPC 码，来避免和/或减少节省存储器的 LDPC 译码器（例如本申请中描述的那种译码器）的延迟。

本申请的发明人认识到 LDPC 译码器中校验节点计算具有出去的可靠性只取两个值的特点，这两个值之一只沿着一个边出去，可以利用这个特点来有效地储存校验节点消息信息，并且有效地实现校验节点处理器模块。利用这个特点，在校验节点处理过程中，不需要储存每个边的全部消息，将存储量减少到已经完成和节点对应的校验节点处理的时候，处理和构建输出消息所必需的量。通过利用消息压缩，在许多情况下，和不按照本发明使用消息压缩的实现方式相比，能够显著地降低对校验节点压缩器模块的存储要求。

本发明的各特征涉及校验节点处理模块。这些处理模块储存和消息对应的压缩格式的消息信息，这些消息和每个校验节点相联系。为此，采用例如每个校验节点包括一个条目的校验节点状态存储器。校验节点状态信息包括从这个特定校验节点的输入消息，而不是这个校验节点的完整一组消息，产生的信息。因此，状态信息代表一组压缩

的消息信息。收到每一则输入消息（例如变量节点到校验节点消息）的时候，更新和每个校验节点对应的压缩信息。可以按任意顺序接收输入消息，例如不必连续接收和处理每个校验节点的输入消息。因此，根据本发明，可以处理和一个校验节点对应的输入消息，接下来是另一个校验节点对应的输入消息，而不是首先接收另一个校验节点的全部输入消息。这样就能够按照变量节点边顺序接收和处理变量到校验节点消息，这和边出现在 LDPC 图的校验节点侧的顺序不同。因此，假设按照变量节点顺序在变量节点处理单元中处理消息，就不需要在本发明的校验节点处理模块处的处理之前，对产生的消息重新排序。

在处理了和单个校验节点相对应的完整的一组变量节点到校验节点消息以后，对和校验节点相联系的完整的一组消息所对应的压缩消息信息进行访问和处理，例如进行解压缩处理，在这里也叫做提取处理。提取处理产生将由单个校验节点产生的完整的一组校验节点到变量节点消息，例如，如同所实现的特定码结构，以及这种码结构内单个校验节点的位置所决定的一样。

在一些实施例中，为每个校验节点储存的状态包括两个值：例如第一和第二值。这些值可以是消息幅度值。状态也包括和这些值之一相联系的边位置信息。这一信息代表产生上述特定校验节点产生的出去的消息中的幅度部分所使用的状态。除了代表可靠性信息的消息幅度信息以外，为每个校验节点储存一个积累符号位。积累符号位由每个输入消息的符号位和利用上次产生的积累符号位值处理的校验节点进行 XOR 产生，以便为每个输出消息产生所述符号位值。

在一个特定的实施例中，为收到的对应于一个校验节点的每个输入消息储存另外的符号位信息。在这样一个实施例中，储存来自在校验节点边上收到的每个消息的输入符号位。因此，在这样一个实施例中，除了每个校验节点的积累符号位以外，为每个校验节点边储存输入符号位，其中每个边对应于不同的输入消息。

尽管在产生从特定单个校验节点的输出消息之前，需要处理和上述单个校验节点相对应的完整的一组输入消息，但是通过采用在译码过程中校验节点不从晚很多处理的变量节点接收输入消息的码结构，

可以为至少一些校验节点产生输出消息而不必处理来自所实现的图结构中存在的完整的一组变量节点的所述消息。通过采用一个码，能够减少与校验节点输出消息的产生相联系的时序延迟和/或使其最小化，其中所述的码考虑了在图的一部分进行校验节点处理的好处，因此它们不会依赖于来自变量节点，将会晚很多处理的消息。

一处理完完整的一组输入消息（例如每个校验节点边一个输入消息），就可以产生校验节点的校验节点输出消息。为了产生校验节点输出消息（它充当变量节点的输入消息），本发明的校验节点处理模块读取和校验节点相联系的校验节点状态。从储存的幅度状态信息，例如第一和第二幅度值以及边标识符，校验节点处理模块对状态进行处理，以便为一个边提取（例如产生）完整的输出消息，例如校验节点到变量节点消息。对校验节点的每一个边重复这一处理，直到产生完整的一组输出消息。在效果上讲，校验节点状态信息的处理是解压缩操作。

在一些实施例中，用最小和算法储存压缩状态的校验节点消息幅度（可靠性）。在这些实施例中，为每个校验节点储存最小消息幅度值、第二小消息幅度值和表明最小消息幅度值对应的边的信息。这一信息是除了积累的符号位信息以外的信息。每次收到与所述信息所对应的特定校验节点相对应的输入消息的时候更新这一信息，直到已经处理完特定校验节点的每一则输入消息。

为了产生和边对应的出去的消息的幅度部分，将边和储存的表明收到最小消息值的边的边标识符进行比较。如果边（正在为这个边产生出去的消息）和储存的边标识符不匹配，就把最小消息值用作出去的消息的幅度。如果储存的边标识符和边（正在为这个边产生出去的消息）相匹配，就把第二小值用作出去的消息值。因此，校验节点产生的出去的消息的幅度要么具有最小值，要么具有第二小值，这个第二小值在提供校验节点收到的最小幅度值的那个单个边上输出。

从计算的角度说，这样就能够在两个可能可靠性 A 和 B 之间进行多路复用（选择）操作，其中 A 和 B 是两个可能的出去的幅度值，例如特定校验节点收到的最小或第二小收到的消息幅度值。如果边

(变量到校验节点连接) 和储存的表明收到幅度值 A (最小值) 的边的边下标相匹配, 就为出去的消息的幅度部分选择第二个出去的幅度值 B。否则输出最小幅度值 A。

在一个特定的示例性实施例中, 可以用各种方式产生出去的消息的符号位。和校验节点对应的积累的符号位值与储存的收到的符号位值进行合并 (例如 XOR), 这个储存的收到的符号位值和一个边相对应, 正在为这个边产生出去的消息, 以便为这个边产生出去的消息的符号位值。重复这个过程, 就象每个边 (为这些边产生出去的消息) 的出去的消息产生过程中的幅度产生部分一样。

从实现方式的角度看, 本发明的校验节点处理方法具有减少通过使用消息压缩实现校验节点处理所需要的存储器的量的优点。从实现方式的角度看, 这一点非常明显, 特别是在校验节点具有大量边的情况下, 如同今天正在使用, 将来很可能使用的许多健壮的 LDPC 码的情况一样。

例如, 考虑 5 比特消息的情况, 它包括 1 个符号位和 4 个幅度位。在以上假设下, 可以压缩从校验节点输出的信息, 压缩到近似 $K+8+\log_2 K$ 比特。其中 K 是进来的和出去的消息的数量。在这个 5 比特消息实例中, 要用 K 个存储位来储存和 K 个输入消息的每一个相对应的那些符号位, 将会用 8 个比特来储存输出消息可能取的两个可能的幅度值中的每一个, 例如最小输入消息幅度和第二小输入消息幅度值, 用 $\log_2 K$ 比特来表明要接收第二可靠性值的边 (消息), 而其它的边则将接收最小输入消息幅度值。如果 K 很大, 这会是高速率 LDPC 码情形, 那么和储存对应于每个收到的消息的完整位组, 作为产生输出消息的处理的一部分的实现方式相比, 使用这一消息信息存储技术能够充分节省。

尽管和储存完整消息的实施例相比, 从储存的状态 (例如最小消息幅度、第二小消息幅度、和最小消息幅度对应的边位置, 以及储存的和每一个边相对应的那些符号位) 产生出去的消息需要一些其它的处理, 但是, 回过头来, 会充分地节省存储器。

本发明的校验节点处理模块具有以下附加优点: 可以接收和处理

充当校验节点处理模块输入的变量到校验节点消息，而不用考虑具体的消息处理顺序。处理了校验节点的每个输入消息以后，产生输出消息。

通过下面的详细描述，本发明数不清的其它特征和优点将会变得显而易见。

附图说明

图 1 说明包括十个变量节点和五个校验节点的一个示例性的 LDPC 码；

图 2 是图 1 所示 LDPC 码的另一种表示，它用矩阵表示来说明所述码，作为图 1 所示图表示的另一个选择；

图 3 说明按照本发明实现的约束节点处理模块；

图 4 说明按照本发明实现的 LDPC 译码器；

图 5 说明按照本发明使用 N 个并行约束节点和变量节点处理单元实现的另一个 LDPC 译码器；

图 6 说明可以用于控制例如图 4 所示译码器的译码的一个示例性的 LDPC 码结构；

图 7 说明本发明中可以用于控制例如图 4 所示示例性译码器的译码的另一个示例性的 LDPC 码结构；

图 8 说明利用图 4 所示译码器和图 7 所示 LDPC 码结构对一组输入值进行译码操作所得到的结果。

具体实施方式

在这里将下面三个相关申请明确引入作为参考，并且将它们看作本申请的一部分：2001 年 10 月 10 日递交的，发明名称为“METHODS AND APPARATUS FOR DECODING LDPC CODES”的第 09/975,331 号美国专利申请；2002 年 4 月 4 日递交的，发明名称为“NODE PROCESSORS FOR USE IN PARITY CHECK DECODERS”的第 10/117,264 号美国专利申请；以及 2003 年 7 月 11 日递交的，发明名

称为“METHODS AND APPARATUS FOR ENCODING LDPC CODES”的第 10/618,325 号美国专利申请。

本发明各实施例涉及方法和装置，这些方法和装置提供 LDPC 译码器体系结构的简单实现，例如，硬件不那么复杂。本发明的方法和装置充分利用了如下事实：对于特定的算法，从校验节点发送出去的可靠性取两个可能值之一，这些值之一只是沿着一个信息边发送。例如，最小和算法就是这种情况。能够用于以压缩形式储存 LDPC 消息信息的其它算法（例如与最小和算法共享同样特性的那些算法，这种同样的特性是从校验节点发送出去的可靠性只取两个可能的值，这些值之一只是沿着连接到校验节点的一个消息边发送出来）中也存在同样的条件。

根据本发明，和用于控制译码的 LDPC 码结构中的每个校验节点相联系的是状态 S 。这一状态包括当前译码迭代中用于产生出去的消息的进来的消息的可靠性（消息幅度）信息。假设已经结合了消息 m_1, \dots, m_k ，令 S_k 表示状态。然后，给定状态更新函数 G ，在处理了收到的和校验节点相对应的变量到校验节点消息以后得到的给定校验节点的已更新状态可以表示为：

$$S_{k+1} = G(m_{k+1}, S_k).$$

根据本发明的各实施例，状态更新操作是在代表和压缩形式的校验节点相对应的多个消息的幅度部分的状态上进行的。因此，状态更新操作压缩了进来的消息。

在利用 MINSUM 算法来储存压缩格式的消息信息的情况下，储存的消息状态可以是例如 (m_A, m_B, A, s) 形式的。其中 m_A 是到目前为止上述状态对应的校验节点看到的进来的最小可靠性（消息幅度）， m_B 是到目前为止看见的第二小可靠性。 A 表示携带可靠性 m_A 的进来的消息从而表明哪个消息边提供了最小值 m_A 的边， s 是和状态列信息对应的校验节点相对应的进来的消息的符号的 XOR。

通过利用函数 G 更新和校验节点对应的状态信息，并且通过实现译码器，其中可以储存并提取和各个校验节点对应的状态，用于根据要把消息给哪个校验节点利用进来的消息来进行更新，于是，进入

校验节点处理器的消息的顺序基本上可以是任意的。这样，根据本发明，变量到校验节点消息能够按照变量节点顺序到达校验节点处理器模块，按照收到这些消息的顺序处理这些消息。这样就能够按照本发明实现 LDPC 译码器，例如图 4 和 5 所示的译码器 400 和 500，其中分别作为变量节点和校验节点处理器实现的两方（变量节点和校验节点）都按照变量节点顺序更新。更进一步，在一些实施例中，变量节点和校验节点处理单元都可以并且确实以并行方式工作，例如同时工作。

和以下情形中的译码器相比，本发明的这种译码器能够实实在在地减少对存储器容量的要求，这种情形是：按照变量节点消息顺序进行变量节点处理，按照校验节点消息顺序进行校验节点处理，并且用存储器来储存在校验节点和变量节点处理器之间传递的消息，对消息进行重新排序。

已经描述了本发明中 LDPC 译码器的一些基本原理和优点以后，下面将描述实现本发明的一个或多个特征的各种示例性模块和 LDPC 译码器。

图 3 说明本发明中的一个校验节点处理模块 300，也叫做约束节点处理模块。模块 300 通过输入端 302 接收变量到校验节点（V2C）消息，通过控制信号输入端 324 接收控制信息，并且产生校验节点到变量节点（C2V）消息，通过输出端 322 输出。校验节点处理模块 300 储存消息信息，这些消息信息可以被用来产生压缩格式的校验节点到变量节点消息。

校验节点处理模块 300 包括校验节点状态存储器模块 310、控制模块 309、校验节点处理器单元 308、消息符号存储器 312、校验节点状态缓冲存储器 314 和校验节点提取器 316，它们连接在一起，如图 3 所示。在图示实施例中，收到的每个 V2C 消息的符号值和幅度值分开。通过输入端 304 把幅度消息值提供给校验节点处理器 308，而把和收到的每个消息对应的符号值提供给校验节点处理器单元 308，同时储存在消息符号存储器 312 中，该存储器 312 用于储存符号位，以后用这些符号位从储存的状态 321、323 产生出去的 C2V 消

息。

对于用于控制译码的码结构中的每个校验节点，校验节点状态存储器 310 包括单状态存储器单元 321、323，该单状态存储器单元 321、323 用于储存存储器单元所对应的特定校验节点的状态。每个校验节点状态存储器 321、323 都储存和用于控制译码的 LDPC 图结构所对应的一个校验节点相对应的状态信息。

在图 3 所示的实例中，以压缩方式储存状态，该压缩方式对应于上面讨论的最小和算法。在校验节点处理一开始就重置条目，该条目和边消息对应的每次处理迭代的条目相对应，该边消息包括在用于控制译码的 LDPC 图的一个完整表示中。图中从一次迭代到另一次迭代的校验节点的处理可能在第一次迭代完成之前发生，例如和图中一个校验节点对应的完整的一组变量节点一处理完成的时候。在这种情况下，全部校验节点状态信息组 321、323 的重置不会同时发生。相反，和校验节点对应的校验节点状态会被全部更新，然后在提供给校验节点状态缓冲存储器用于产生 C2V 消息以后重置。

每一个状态条目 321、323 对应于用于控制译码的图结构中的一个校验节点。每个状态条目 321、323 包括 S，它是一个一比特值，是收到的每个 V2C 消息所对应的符号位的 XOR，其中的 V2C 消息是给条目所对应的校验节点的；最小值 m_A ，它表明收到的最小消息幅度，这些消息幅度对应于特定的校验节点；第二小消息幅度值 m_B ，对应于上述特定的校验节点；以及下标 I_A ，它表明在上面收到最低的最小值 m_A 的消息边。

控制信号输入端 324 接收控制信号，利用该控制信号，根据用于控制译码的码结构（例如，根据图中边的数量和因此在变量和校验节点处理单元之间传递的消息）来控制校验节点处理模块的工作。控制信号 324 包括信息表明在特定时间在输入端 302 处收到了哪个边，从而收到了哪个 V2C 消息。在 V2C 输入消息时序和期望的 C2V 输出消息时序之间存在固定的或已知的关系的情况下（常常是这种情况），可以用这个同样的信号来驱动 C2V 消息产生。把控制信号 324 提供给控制模块 309、消息符号存储器 312 和校验节点读取处理器（提取

器) 316。控制模块 309 用收到的控制信号来确定收到的 V2C 消息对应于哪一个校验节点。基于收到的控制信息，控制模块确定要从校验节点状态存储器 310 读出哪一组校验节点状态信息 321、323 来进行更新。这样，控制模块基于边信息来确定收到的消息所对应的校验节点。校验节点处理器利用收到的消息信息一更新完所提取的状态信息，如同后面将讨论的一样，就把更新过的状态写回从中提取要更新的状态的校验节点状态条目 321、323。这一过程将继续下去，直到控制模块 309 通知校验节点处理器单元 308，已经在这个特定的处理迭代中，利用给这个校验节点上一个 V2C 消息，全部更新了特定校验节点的校验节点状态。全部更新了校验节点的状态以后，校验节点处理器通过将默认值写入校验节点状态，并且输出全部已更新校验节点状态给校验节点状态缓冲器 314 用于消息提取，来重置校验节点状态中的值。校验节点状态缓冲存储器 314 包括一个或多个校验节点的状态条目，这些条目和条目 321、323 一样具有同样的内容，但是值和这个校验节点的一组全部更新过的状态相对应。根据从输入端 324 获得的控制信号，校验节点状态缓冲存储器 314 知道要储存全部更新过的校验节点信息的条目。

校验节点状态存储器 310 包括已更新校验节点状态消息输入端 327，用于接收要储存的状态信息，还包括控制输入端 327，用于接收控制信号，该控制信号表明要访问哪一组校验节点状态信息 321、323，并且是要把已更新状态信息存入所表明的那组校验节点状态信息，还是要从中读出。通过输出端 329 输出从校验节点状态存储器读取的校验节点状态信息 321 或 323，并且将它提供给校验节点处理器单元 308 的状态输入端，在那里将它用于状态更新操作。

校验节点处理器 308 根据收到的 V2C 消息中包括的信息，对从存储器提取的一组状态信息所进行的状态更新操作为：将收到的 C2V 消息的幅度 m_r 和提取的状态信息中的最小幅度 m_A 进行比较，其中提取的状态信息对应于校验节点，而收到的消息则是给这个校验节点的。

如果 m_r 小于 m_A ，那么：将 m_B 设置成等于 m_A ，以创建已更新

m_B , 并且将 m_A 设置成等于 m_r , 从而使 m_r 成为已更新最小值, 前一个最小值变成校验节点的目前第二小。另外, 表明最小 m_A 所对应的消息边的下标 I, 表明在上面收到正在处理的消息 (例如提供已更新最小幅度值的消息) 的消息边。

如果 m_r 小于 m_B , 但是大于 m_A , 那么将 m_B 设置成等于 m_r , 而不改变 m_A , 也不改变 I。

如果 m_r 等于或大于 m_B , 就不改变从存储器读取的 m_A 、 m_B 和 I 值, 已更新状态将包括从存储器读出的这些单元的值。

不考虑收到的消息值和储存的最小值的相对幅度, 通过对收到的消息中的符号位和读自储存的校验节点状态条目的符号位进行 XOR 操作, 用收到的每个消息更新校验节点的积累的符号位 S。XOR 操作的结果变成校验节点的已更新条目的符号 S 位, 在没有被完全更新的校验节点情况下被写回存储器, 或者在全部更新了校验节点条目的情况下被提供给校验节点状态存储缓冲器 314。如上所述, 在全部更新了校验节点条目的情况下, 会把默认值写入校验节点的条目 321、323, 从这些条目读取校验节点状态, 从而为另一次图形处理迭代重置信息。正常情况下最小 m_A 和第二小 m_B 的默认值会是这些单元的最大值。

给定上述校验节点处理和状态更新技术, 很显然, 校验节点处理器单元 308 能够按照任意顺序处理变量到校验节点消息, 例如, 按照变量节点顺序, 而不是校验节点边顺序。这就使得校验节点处理器单元 308 能够按照 V2C 消息的自然产生顺序处理它们, 作为变量节点处理器操作的结果, 给定变量节点处理器通常都是按照变量节点边顺序进行操作的。

由于校验节点的全部更新了的状态信息是产生 C2V 输出消息所需要的信息的一种压缩格式表示, 因此用另外的处理来构建通过输出端 322 的实际 C2V 消息输出。校验节点读取处理器, 例如提取器, 316 从校验节点状态存储器 314 中储存的全部更新了的状态, 以及存储器 312 中储存的储存符号值, 产生完整的 C2V 消息, 为每个收到的 V2C 消息储存一个符号值 313、315。

这样，和不是以压缩格式储存消息的实现方式比较，“读取”C2V 消息需要额外的处理，以便进行解压缩处理，如果不实用压缩存储格式，就不需要这种解压缩。直接从符号存储器读取储存的进来的消息符号值 313、315。从存储器 312 读取的每个符号值都和对应于校验节点的全部更新了的和 S 进行 XOR 操作，以获得与（和出去的 C2V 消息相联系的）消息边相对应的出去的符号。这样，从消息边上收到的消息符号和对应于 LDPC 图中连接了消息边的校验节点所对应的完整的一组消息的 XOR 的积累符号值 S，产生消息边的出去的符号。从为校验节点储存的状态中包括的最小、第二小和消息下标提取可靠性，例如消息幅度部分。如果读取的边的下标和 A 匹配，就输出可靠性 m_A ，否则输出可靠性 m_B 。表示充当消息边标识符的下标 A 有许多可能性。尽管简单的边号是一种表示下标 A 的简单方式，但是也可以使用其它的编号方案和/或边下标编制方法。

消息提取器 316 产生的消息幅度在幅度输出端 318 上输出，而出去的消息符号则在输出端 320 上输出，将它们组合起来形成通过输出端 322 的完整的消息输出。

参考图 3 描述了本发明的校验节点处理模块 300 以后，下面将参考图 4 描述这个模块在本发明一个示例性的 LDPC 译码器实现方式中的使用。

图 4 画出了用图 3 所示校验节点处理模块 300 实现的一个示例性 LPDC 译码器 400。译码器 400 按照变量节点边顺序进行变量节点处理和校验节点处理。除了校验节点处理模块 300 以外，译码器 400 包括译码器控制模块 402、变量节点处理单元 404、软输入缓冲器 406、软/硬输出缓冲器 412、约束校验节点 408 和迭代控制逻辑电路 410。

在载入变量节点处理单元 404 之前，把要译码的输入值提供给软输入缓冲器 406。译码器控制模块 402 负责按照储存的和控制译码操作使用的 LDPC 图结构相对应的译码控制信息，产生译码器控制信号。译码器控制模块产生控制信号，这些控制信号用于控制参考图 3 所讨论的校验节点处理模块操作，以及变量节点处理单元操作。在译码器控制模块的控制下，变量节点处理单元顺序载入要处理的输入数

据的那些部分，例如包括幅度信息和符号位的收到的消息值。在初始迭代过程中，没有任何已有的约束消息供变量节点 PE 404 处理，通过处理所述输入数据来产生 V2C 消息。产生 V2C 消息，并且按照变量节点消息边顺序从输入数据输出。产生的包括幅度和符号值的 V2C 消息被提供给校验节点处理模块 300 的 V2C 输入端 302。消息还被提供给约束校验模块 408，而软/硬输出乒乓缓冲器则接收和产生的每个 V2C 消息相联系的符号位。约束校验模块 408 进行校验，以确定和译码器图的一次迭代相对应的收到的消息符号值，是否满足预定译码约束，例如奇偶校验。在已经产生了 V2C 和 C2V 消息一个完整组的消息传递迭代的结尾，约束校验模块 408 确定是否满足奇偶校验。如果检测到当前消息传递译码迭代已经导致了成功译码，模块 408 就产生译码完成信号，或者，如果还没有确定译码已经成功，就产生迭代完成信号。约束校验模块 408 产生的信号被提供给迭代控制逻辑 410。在每一次迭代结尾，假设还没有满足约束校验，储存的和图的边相对应的符号值可以从输出缓冲器 412 作为软判决输出，或者在约束校验已经满足的情况下作为硬判决输出。

在预选数量的消息传递迭代没有导致满足奇偶校验信号以后，迭代控制逻辑 410 将发出译码失败信号，作为超时信号，或者如果在超时之前满足了奇偶校验，就发出译码成功信号。这样，控制逻辑 410 包括一个超时功能，如果在预选次数译码迭代以内没有成功完成译码，这个功能就终止译码。

LDPC 译码器 400 按照变量节点边顺序进行变量和校验节点处理操作，因此，和其它系统相比，减少或消除在变量节点和校验节点处理单元 308、304 之间进行消息存储的需要，在这些其它系统中，校验节点处理是按照校验节点边顺序进行的，变量节点处理是按照变量节点边顺序进行的。此外，如同下面将讨论的一样，通过仔细地选择 LDPC 图结构，可以在 LDPC 图结构的每次迭代的处理中存在重叠，而不需要提供完整的一组重复的约束节点状态存储器条目 321、323。

使用译码器 400 的译码迭代可以按照以下方式进行。一次更新一个变量节点。从校验节点输出状态存储器和符号存储器读取消息，产

生出去的校验节点消息。在标准变量节点处理之后，这些消息在变量节点里加起来，随后从这个和中减去。产生出去的消息的时候，将它们直接发送给校验节点处理器，该处理器还从存储器提取对应的部分状态。更新这一状态，并且将它返回到部分状态存储器。如果 V2C 消息是当前迭代中这一约束的最后一个，就可以将这一状态写入输出状态存储器，并且重置部分状态存储器。

按照本发明，可以很容易地采用本发明的校验节点处理模块 300 和图 4 所示的通用 LDPC 译码器系统，来支持多个约束节点和变量节点处理单元的使用，这些节点处理单元并行布置并且并行工作。在图 5 中，译码器 500 并行地使用 N 个约束节点，约束状态提取电路，以及变量节点处理单元。在图 5 中，约束校验模块 308'、迭代控制逻辑 310'以及缓冲器 312'和 306'跟参考图 4 所描述的类似编号的单元一样，按照相同或相似的方式工作。校验节点处理模块 300'中的其它单元也和图 4 中相似地编号，但是不带'号的单元一样，按照相同或相似的方式工作。然而，在图 5 所示的实施例中，单元 308'包括并联排列的 N 个约束处理器 308，而约束节点状态存储器 324 也被设计成和校验节点状态组数的 N 倍工作，同时约束状态提取模块 316'包括 N 个消息提取电路。注意，在支持一次访问 N 组状态信息的时候，和图 4 实施例相比，存储器 310'不必包括任何其它状态条目，因为状态条目的数量由图中校验节点的数量决定，而和有多少节点是并联的无关。

图 5 所示的实施例包括译码器控制模块 502，它被设计成基于小图描述和关于如何修改小图来产生用于控制译码的更大图的信息，支持 N 宽译码操作。对小图的修改可以按照变量节点和校验节点处理单元之间传递的消息的重新排序（例如转动）来实现，其中 N 个消息的组是并行地传递的。特别是，在图 5 所示的实施例中，当消息在译码器 500 的变量单元之间传递的时候，用开关 530、526、528 来控制消息的重新排序。重新排序是根据控制逻辑 518 中储存的图描述信息和置换图 522 产生的转动信号来进行的，作为控制逻辑 518 驱动的列计数器 520 的函数。用于控制提供给校验节点处理模块 300'的消息

的重新排序的转动控制信号是通过使用延迟线 524 进行延迟来产生的，一些转动信号用于控制变量提供给变量处理单元 504 的输入端的消息的重新排序。

描述了校验节点处理器模块 300，并且按照本发明实现了各种 LDPC 译码器 400 和 500 以后，下面将讨论关于图设计的各种特征，LDPC 码结构，以及实现本发明的译码器中，用于特定图结构的译码实现。

为高速应用实现图 5 所示的译码器的时候，有时叫做投影图的储存的图描述信息所描述的小图，通常都特别小，也就是说，它会有少量变量节点。如同所描述的一样，这种体系结构每个时钟周期进行一次投影的边更新，它对应于每个时钟周期中处理 N 个消息。如果完整的图迭代的译码必须在能够开始下一迭代之前完成，就要在译码系统中引入延迟。在长管线延迟（例如大图情形）情形中，这样做会降低效率。管线延迟是在每个边已经启动处理以后，完成对每个边的译码操作所需要的附加时钟周期。如果管线延迟比投影的图尺寸大，那么这一附加延迟就会显著地降低译码器的速度。

但是，通过明智的图设计，可以部分地解决延迟问题，不管它是在译码器 400 中发生，还是在并行译码器 500 中发生。

假设所述组已投影变量节点 V 可以分成多组，例如至少分成 SV1 和 SV2 这两组，其中 SV1 在 SV2 前面，所述组已投影校验节点 C 可以分成至少两组 SC1 和 SC2，从而使 SC1 中的校验节点通过图中的边只和 SV1 中的变量节点连接。然后，如果按照变量节点顺序进行处理，对应于变量节点 V1 的消息将首先处理，然后是对应于变量节点 V2 的消息，接下来是来自变量节点 V3 的消息，如此下去。在这种情况下，如果 SC1 中的最后一个校验节点不和 SV2 中的最后一个变量节点的最后边相连接，在对应于 SV1 中最后一个变量节点的最后消息已经被处理完的时候，对应于第一组校验节点 SC1 的消息的处理将已经完成，或者已经在此之前完成。在为下一个图迭代对来自变量节点 V1 的 V2C 消息的处理开始的时候，得到的对应于 SC1 组中的校验节点的全部完成的状态可以储存在校验节点状态存储器中，

用于产生 V2C 消息。这样，图处理的下一次迭代能够开始，而约束节点处理模块 300 则仍然在处理和图处理的未完成迭代相对应的 V2C 消息。

这是因为在第 $n+1$ 次迭代中，来自变量节点组 SV1 的消息的处理开始的时候，C1 中的约束已经在第 n 次迭代中全部更新。约束这个图让它具有这种类型的划分允许随后的迭代在时间上迭代，并且能够有效地减少和/或消除因为管线延迟带来的不利后果。

在图 6 中说明具有这种特性的 LDPC 码和对应的图结构 600。在图 6 中，这个图包括 8 个校验节点 602，标记为 $C_1 \sim C_8$ ，以及 16 个变量节点 604，标记为 $V_1 \sim V_{16}$ 。注意，到来自变量节点组 SV1 中的变量节点 $V_1 \sim V_{12}$ 的消息由约束节点处理器模块 300 处理的时候，对应于前 4 个约束节点 $C_1 \sim C_4$ 的状态将已经全部更新。全部更新的这个状态可以被传递给约束节点缓冲存储器 314，释放出和这个图处理的下一次迭代要使用的约束节点 $C_1 \sim C_4$ 对应的约束节点条目。这样，作为 SC2 这一组中约束节点的处理，也就是说，约束节点 $C_5、C_6、C_7$ 和 C_8 继续一次迭代，将会产生和 SC1 组中的节点相对应的迭代的 C2V 消息，同时下一次迭代的约束节点处理能够开始重叠与当前迭代相联系的继续处理。

本发明的 LDPC 译码器方法和装置特别适合于高速率码，其中校验节点数常常充分小于图中的边数。在这种情况下，按照本发明使用的状态存储器会倾向于充分地小于边存储器，否则就会需要，如果储存了全部消息用于约束节点处理。

图 7 是一个较简单的图，将会把它用于解释译码器处理，例如，将图 4 中的译码器 400 用于对具有消息值 0、-1、5 和 4 的一组示例性的输入数据进行译码的时候会进行的处理，在这个图的第一次处理迭代过程中将这些消息值载入变量节点。

在图 7 中，图 700 包括按变量节点顺序从 0 到 8 编号的 9 边连接在一起的总共三个校验节点 702 和四个变量节点 706。注意，如果从校验节点一侧给边编号，那么这些边会有不同的编号顺序，因为它们出现在校验节点中的顺序不同于变量节点的顺序。在变量节点 706 和

校验节点 702 之间沿着图示的边来回传递消息。校验节点 702 包括第一到第三校验节点 710、712、714，而变量节点 706 则包括第一到第四变量节点 720、722、724、726。

在图 800 中示出利用译码器 400 和图 7 所示的码结构得到的各个值。图 800 说明两次完全译码器处理迭代的结果，它们在用于将收到的消息（0、-1、5 和 4）载入译码器 400 的初始迭代以后，一开始处理所述消息。在第二输入值的前面用一个负号来表明负符号位，等效为一个 1 比特，与所述消息相联系，而没有任何负号则表明是正符号位，等效为一个 0 比特。这样，第二消息 -1 具有负符号位，而第一、第三和第四输入消息则具有正符号位。在第一处理迭代的结尾，对应于校验节点 C1、C2 和 C3 的状态存储器条目会包括那组状态存储器条目值 802 中图 8 顶部所示的值。

迭代从提取 V2C 消息开始。在初始迭代中，将提取的 C2V 消息设置成 0。在变量节点处将这些消息加起来，作为当前迭代的一部分，并产生 V2C。这些消息由更新与当前迭代相联系的校验节点状态的校验节点处理器收到。当所有校验节点状态已经被更新的时候，迭代完成。

在那组状态存储器条目值 802 下面的图中，每一行都对应于和不同消息边相联系的处理，例如在提取 C2V 消息的时候，要沿着所表明的边发射的 V2C 消息的产生，以及与给定边相联系的校验节点状态的更新。图的每一次迭代都涉及图 7 图的每个边的一个 C2V 消息的产生和处理，以及每个边的一个 V2C 消息的产生和处理。

一般而言，810、812、816、818 和 820 这些列说明利用前面的译码器迭代中的处理的结果的 C2V 消息提取。822 这一列说明了变量节点和，这个和是所有进来的到变量节点的消息与变量节点所联系的收到的节点的和。

824 这一列说明会沿着这一迭代中的对应边产生的 V2C 消息。它等于变量节点和减去进来的 C2V 消息。这些 C2V 消息将被约束节点处理器收到，并且用来更新 826、828、830 这些列所示的状态信息。826、828、830 这些列说明在当前处理迭代过程中，如何响应收到 824

这一列所示的每个 V2C 消息来更新校验节点处理器状态。

信息 810 说明在从约束节点缓冲器状态存储器 314 读取的前面的图处理迭代以产生 C2V 消息的过程中所产生的校验节点状态。和前一次处理迭代过程中产生的每个校验节点相对应的这个全部更新过状态被用于产生 C2V 消息，这些 C2V 消息由当前图迭代中的变量节点处理器 304 进行处理，以产生 V2C 消息。812 栏说明和沿着一个边从前一次迭代来的 V2C 消息相对应的符号位，从符号位存储器 312 把它取出来用来构建当前处理迭代中的 C2V 消息，这个 C2V 消息会成为变量节点处理器的输入，这个变量节点处理器会为当前处理迭代产生 V2C 消息。814 栏说明在前一次迭代中产生的以及从（对应于和 C2V 消息相联系的消息边所对应的校验节点（C1、C2 或 C3）的）状态获得的，通过将来自以前收到的 V2C 消息的储存的符号位 812 和储存的积累符号位 S 进行 XOR 运算提取出来的约束状态产生的已更新符号位。816 栏表明在前一次处理迭代中约束节点状态提取器进行校验，以确定是应该输出第一小 mA 还是第二小 mB 的结果。这一校验要比较正在产生的 C2V 消息的边下标和对应于储存的第一小的边下表 I。在这里“Y”表示“是”，意思是边下标确实和前一次迭代中发射最小幅度的边相对应，“N”表示“否”，意思是边下标不和前一次迭代中发射最小幅度的边相对应。812 栏说明选择 m_A 或 m_B ，用作出去的 C2V 消息的幅度的选择结果。820 栏说明要在所表明的边上发射的，供当前迭代中变量节点处理器 304 使用的实际 C2V 消息。变量节点处理器 304 产生在处理迭代过程中收到的 C2V 消息的和。然后用 822 栏列出的结果 Vnode 和来产生 V2C 消息，沿着消息边提供给约束处理器模块 300，例如用于在当前迭代过程中进行约束节点处理。

从约束节点处理模块的角度，810、812、814、816、818、820 和 822 栏列出的信息代表从前一次图处理迭代中的计算结果得到的操作。如上所述，假设进行了仔细的图设计，在为一个图处理迭代发生的这些操作时间，和为约束节点处理的下一次迭代的消息状态更新的时间，存在一些重叠。

824 栏说明收到的，要在第二和第三约束节点处理模块迭代过程中处理的 V2C 消息，对应于使用图 7 所示图结构对示例性的输入进行处理。每个图迭代都要对九个 V2C 消息进行处理，这九个 V2C 消息对应于在边 E0~E8 上传递的消息。约束处理模块 300 的消息处理按照变量节点边顺序进行。在处理开始的时候，对应于校验节点的约束存储器将已经重新初始化。假设在处理迭代中没有任何重叠，校验节点 C1、C2 和 C3 中每一个的状态将会在收到对应于边 E0 的第一 V2C 消息之前已经重新初始化。在它已经用收到的 V2C 消息更新以后，其中每一行对应于不同 V2C 消息的处理，826、828、830 栏中的每一栏都分别列出了分别和校验节点 C1、C2、C3 之一相对应的储存的状态。响应特定消息（例如这一行所对应的边上传递的消息）更新的状态内容用粗体字示出。在 826、828、830 栏中使用 X 来说明已经设置成初值的那些值。那些值是“不关心”值，因为在产生的 C2V 输出消息中不会使用它们。

注意，在第二次迭代的末尾，全部更新的消息状态 C1、C2、C3 将会已经更新成如下值：对于 C1 ($m_A = 0, m_B = 4, I = 2, S = 0$)；C2 ($m_A = 1, m_B = 3, I = 1, S = 1$)；并且 C3 ($m_A = 1, m_B = 3, I = 3, S = 1$)。在第三次迭代的末尾，全部更新的消息状态 C1、C2、C3 将会已经更新成如下值：对于 C1 ($m_A = 1, m_B = 3, I = 2, S = 0$)；C2 ($m_A = 0, m_B = 3, I = 1, S = 0$)；并且 C3 ($m_A = 3, m_B = 3, I = 3, S = 0$)。

尽管在和校验节点相对应的消息处理已经完成以后，没有将状态示为被清除，但是在图处理迭代之间存在重叠的实施例中，和已全部更新的约束节点相对应的状态的重置将会储存在约束节点状态缓冲存储器 324 中，并且状态信息在约束状态存储器中重置。在图 8 所示的实例中，没有简单地示出这样的重置，以帮助理解约束状态更新顺序，以及在和完全图迭代相联系的处理过程中如何产生更新状态的一个完全组。

给定能够使用本发明的译码器和译码方法的各种码结构，关于在处理迭代之间能够发生的重叠的多少，有很大的灵活性。为了减小译码延迟，关于进行和不同译码器迭代相对应的校验节点处理的重叠，

可以在图处理迭代中将码结构选择为允许 10%、20%、30%、40%或更多的重叠。

下面描述各种示例性的译码方法实现方式和示例性的变化。

进行低密度奇偶校验译码操作的一种特定示例性方法包括以下步骤：在存储器中储存和多个校验节点收到的消息相对应的消息状态信息；接收给所述多个校验节点之一的校验节点输入消息；基于要将所述收到的消息给到的校验节点，选择用于状态更新操作的所述多组状态信息之一；从所述存储器提取所选择的那组消息状态信息；以及根据所述收到的消息更新所选择的那组消息状态信息。该方法还包括以下步骤：将所述已更新的那组消息状态信息写入从中提取过所述消息状态信息的存储器位置，并且针对在第一序列中收到的多则消息中的每一则，顺序重复所述接收、选择、提取和更新步骤，该第一序列对应于和所述收到的消息对应的边连接到代表 LDPC 图的图中的变量处理节点上去的顺序。在一些实现方式中，和对应于所述收到的消息的边连接到代表所述 LDPC 码的图中的变量处理节点上去的顺序相对应的所述序列，不同于第二序列，对应于所述收到的消息的边按照这个第二序列连接到代表所述 LDPC 码的图中的约束处理节点上去。该示例性的方法还包括：对和校验节点相对应的至少一组状态信息进行校验节点到变量节点消息提取操作，已经为所述校验节点收到了完整的一组变量到校验节点消息，其中多次进行所述提取操作，产生多个校验节点到变量节点消息，连续产生给所述多个变量节点中的单独一个的校验节点到变量节点消息，以产生给所述多个变量节点中的所述单独一个的校验节点到变量节点消息序列。

在所述示例性的方法中，在全部已更新状态被储存在缓冲存储器中，用于 C2V 消息提取处理之前，为每个校验节点产生全部已更新状态。全部已更新状态是已经被和状态对应的校验节点相对应的全部那组 V2C 消息所更新的状态。在和整个图相对应的处理过程中，要更新多组状态。通常要进行和图对应的多次译码迭代，每一次迭代都和代表用来控制译码的 LDPC 码的图中整个一组消息的处理相对应。可以和同一个时间周期中不同的图处理迭代相对应，更新校验节点状

态，其中假设用于后续迭代的校验节点状态已经在前一次迭代过程中全部更新。

这样，在一些实现方式中，该方法包括在译码器处理的所述第一次迭代过程中，在所述至少一个其它校验节点（例如图中所述最后一个校验节点）的所述状态更新全部完成之前，更新和所述第一校验节点相对应的另一组状态，作为变量到校验节点消息处理的第二次迭代的一部分。

作为支持图处理迭代重叠的译码方法的一部分，在一些实现方式中，所述的方法还包括缓存和所述第一校验节点相对应的所述已全部更新的状态；以及从所述缓存的已全部更新的状态提取校验节点到变量节点消息。在一些实现方式中，从所述缓存的已全部更新的状态提取校验节点到变量节点消息的所述步骤包括从所述已全部更新的状态以及和用于产生所述已全部更新的状态的多个变量节点到校验节点消息相对应的储存的符号信息，产生多个出去的消息。

一些示例性的译码方法包括在完成和第二组校验节点相对应的状态更新之前，全部完成第一组校验节点的状态更新，针对和一个校验节点相对应的多个消息边中的每一个将该校验节点的状态更新了一次的时候，该校验节点的状态更新全部完成。在一些实施例中，所述第一组和第二组校验节点中的每一组都包括代表所实现的用于控制译码的 LDPC 码结构的 LDPC 图中校验节点总数的至少 20%。

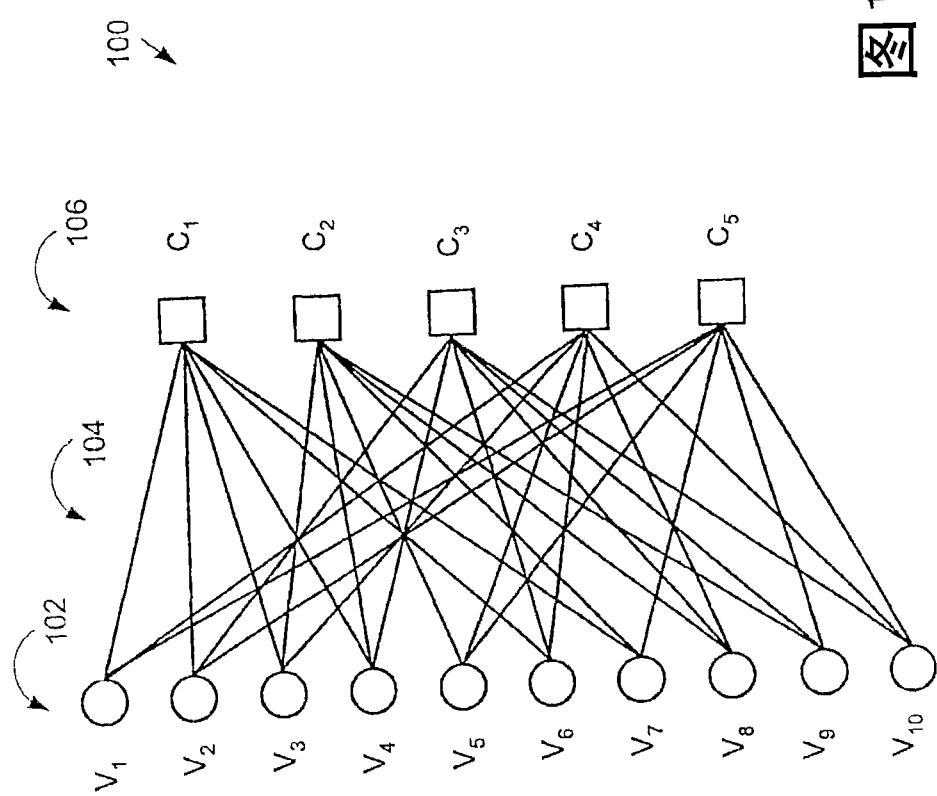
所述状态更新操作可以作为和第一组校验节点相对应的状态更新的一部分进行。在一些实现方式中，第一组校验节点的所述更新是利用第一组变量到校验节点消息在第一时间周期中进行的，该方法还包括：在第二时间周期中更新和第二组校验节点相对应的状态信息，所述第二组校验节点只包括所述第一组校验节点中不包括的校验节点，所述第二时间周期跟随在所述第一时间周期之后。在这些实现方式中，可以在不同的时间提取校验节点信息。在一个实现方式中，所述方法包括在所述第二时间周期中，从和所述第一组校验节点相对应的已更新状态信息提取校验节点到变量节点消息。在实现这一时序功能的一些实现方式中，所述第一和第二时间周期在长度上相等。所述

第一和第二时间周期可以并且常常被第三时间周期分开，在该第三时间周期中更新和第三组校验节点对应的状态。常常是使用包括许多节点的大图。在一些实现方式中，所述第一和第二组校验节点中的每一组都包括和用于控制译码的 LDPC 码相对应的图中校验节点的至少 10%。在一些实现方式中，所述第一和第二组校验节点中的每一组都包括和用于控制译码的 LDPC 码相对应的图中校验节点的至少 20%。在一些实现方式中，所述第一时间周期少于处理 N 个变量到校验节点消息所需要的时间的 40%，其中 N 等于和用于控制译码的 LDPC 码相对应的图中消息边的总数，而在其它实现方式中，所述第一时间周期少于处理 N 个变量到校验节点消息所需要的时间的 20%，其中 N 等于和用于控制译码的 LDPC 码相对应的图中消息边的总数。

在本发明的范围之内，上面描述的方法和装置仍然有数不清的变化。例如，在使用关于消息的压缩状态的时候，可以按照和上面描述的方式略为不同的方式为 C2V 消息产生符号信息，同时仍然保持在本发明的范围之内。另一种实现方式采用乒乓结构，其中将一个存储器用作状态存储器，而另一个则包含前一次迭代的全部状态信息，并且在一次迭代的结尾，将这些缓冲器的角色翻转。储存了 V2C 消息的符号。除了可靠性信息以外，还维护进来的符号位的 XOR，作为状态的一部分。提取 C2V 消息的时候，将储存的 V2C 符号和状态的积累 XOR 进行 XOR，以得到 C2V 消息符号位。注意，一读出它，就不再需要 V2C 符号位，于是马上就可以随后被变量节点处理器产生的 V2C 符号位覆盖它。

上述方法可以用计算机系统实现，该系统包括存储器、CPU 和连接在一起的一个或多个输入和/或输出装置。在这样一个实现方式中，存储器包括按照本发明实现的例程。执行的时候，这个例程让 CPU 按照本发明接收、处理（例如进行译码）和输出数据。另外，本发明的步骤还可以用专用硬件（例如电路和/或硬件和软件的组合）实现。

图1



$$\begin{array}{c} \text{206} \\ \curvearrowleft \\ \boxed{x^{-} x^2 x^3 x^4 x^5 x^6 x^7 x^8 x^9 x^{10}} \\ x = \\ \text{202} \\ \curvearrowleft \\ \boxed{\begin{matrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{matrix}} \\ H = \end{array}$$

图2

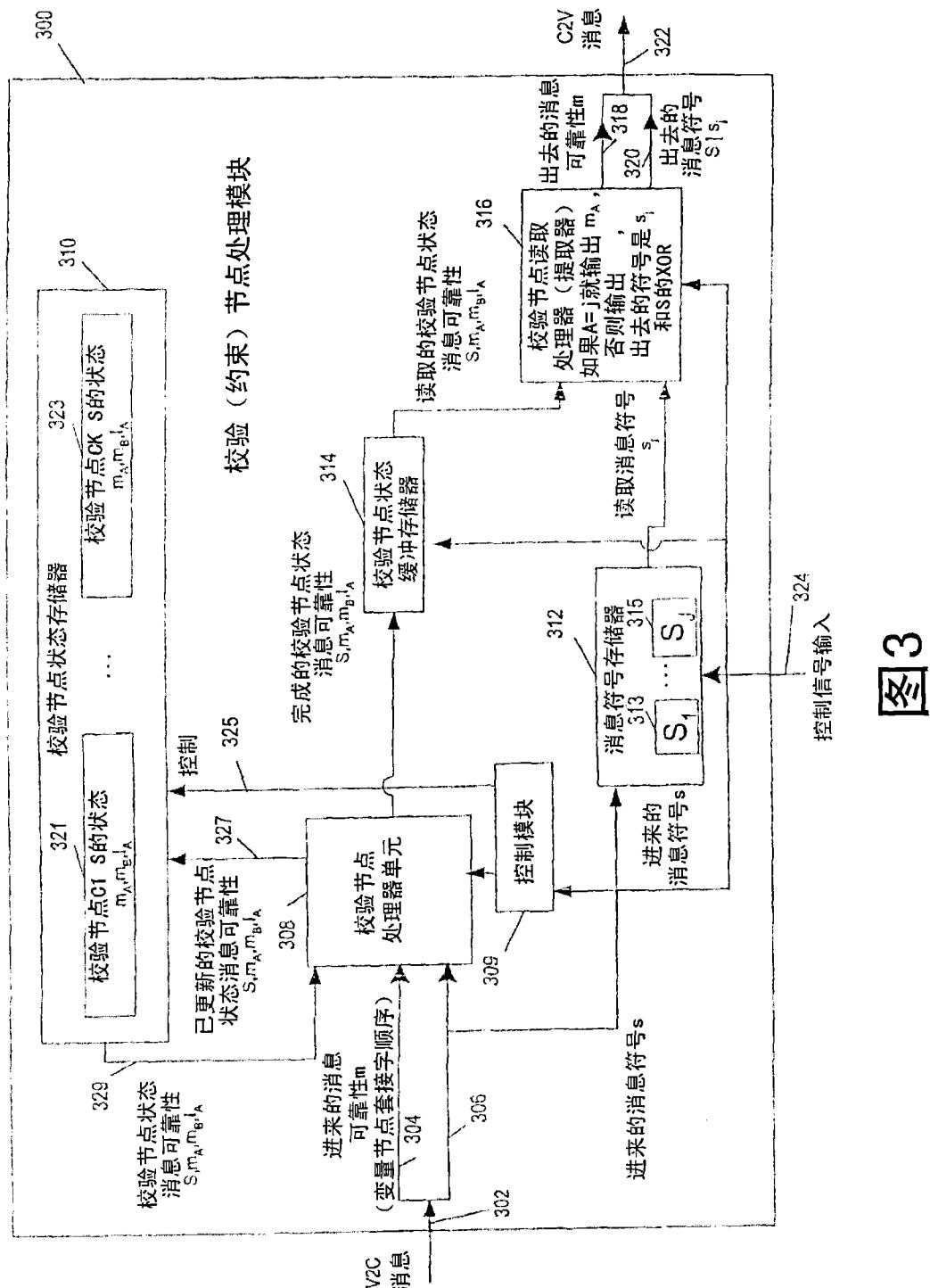


图 3

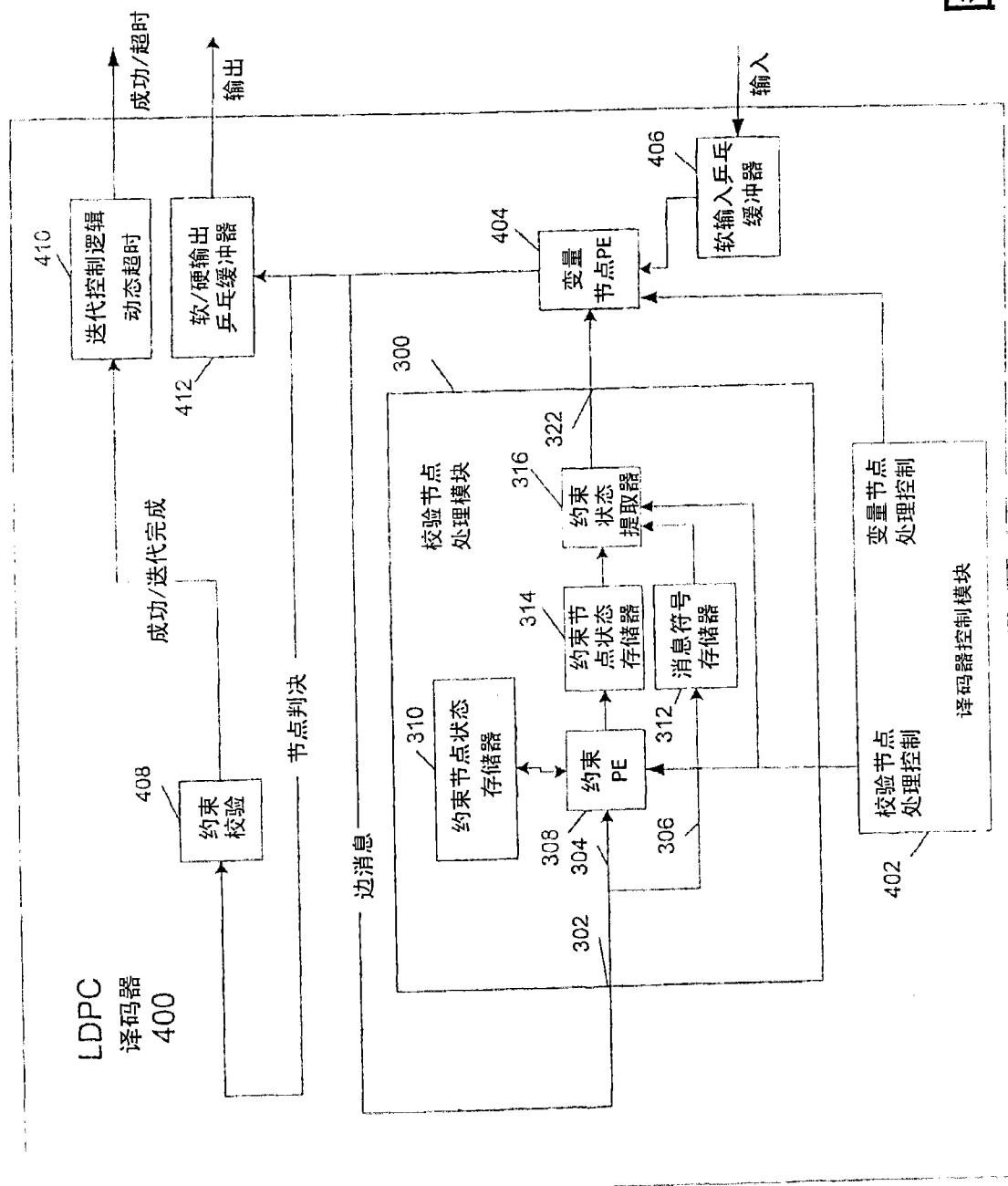


图5

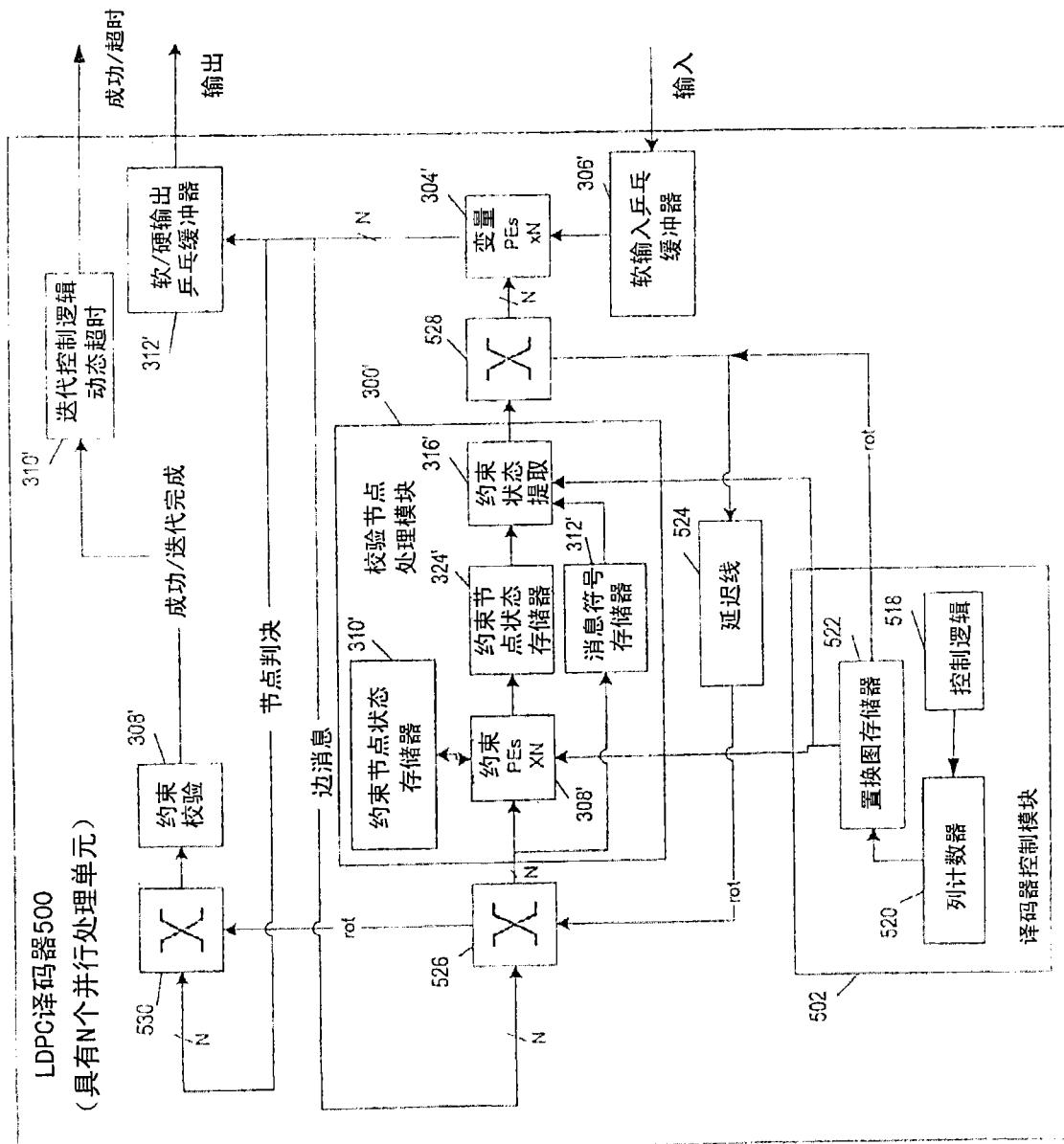


图6

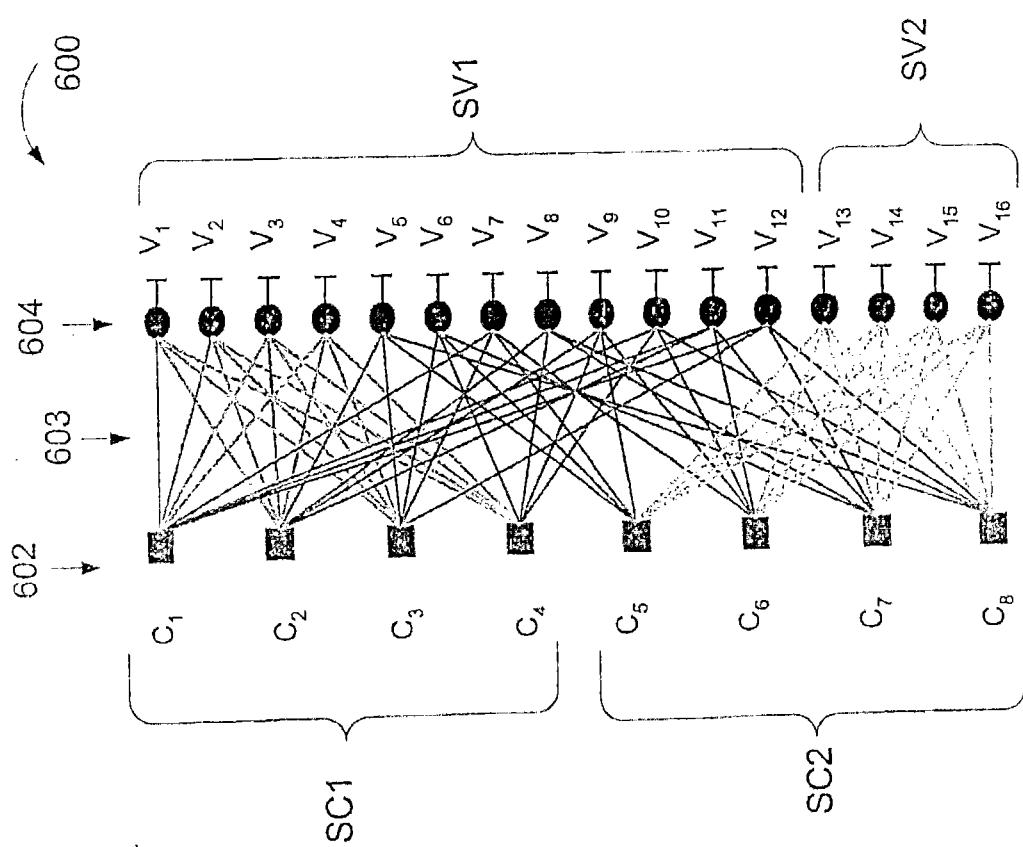
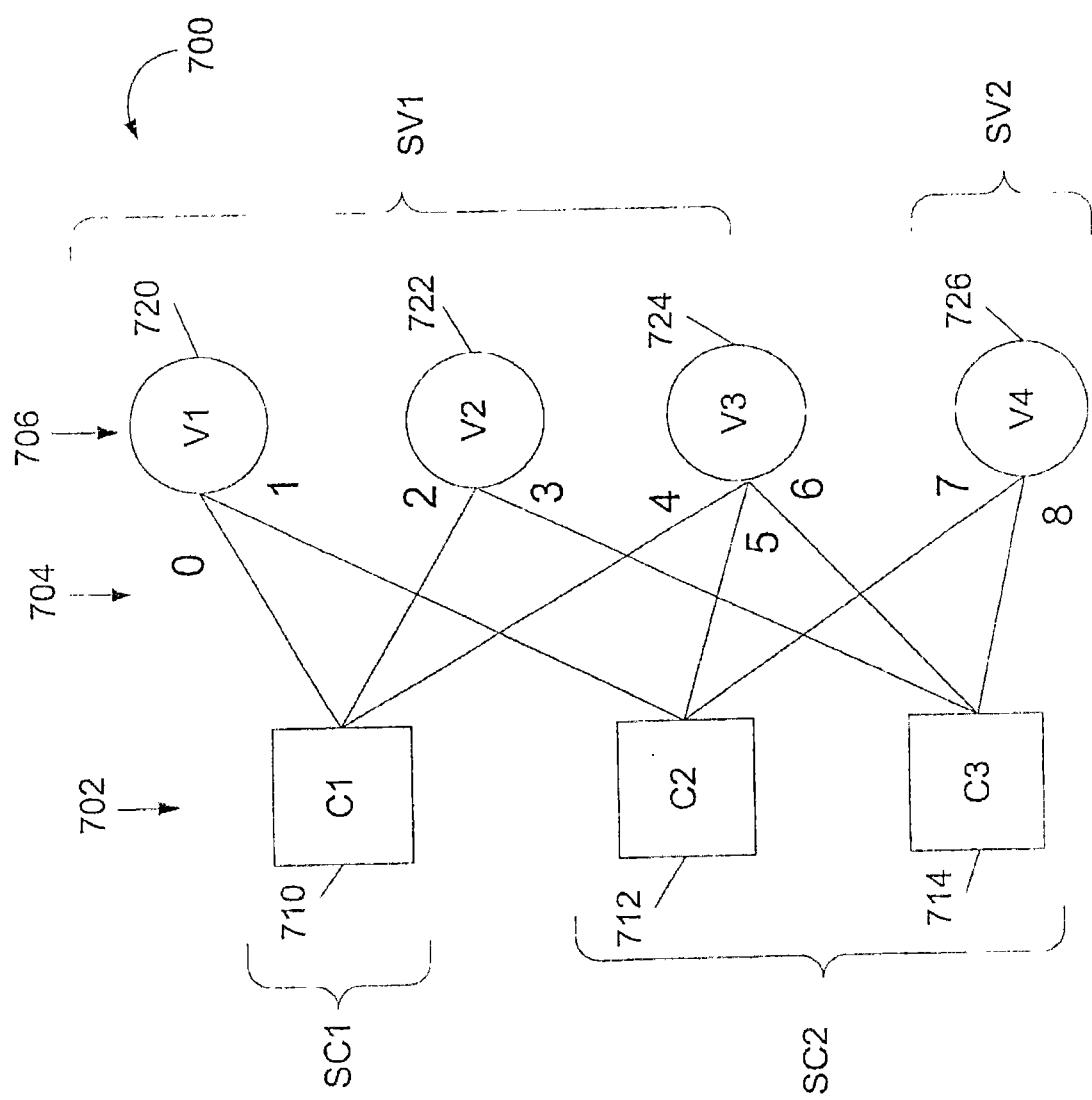
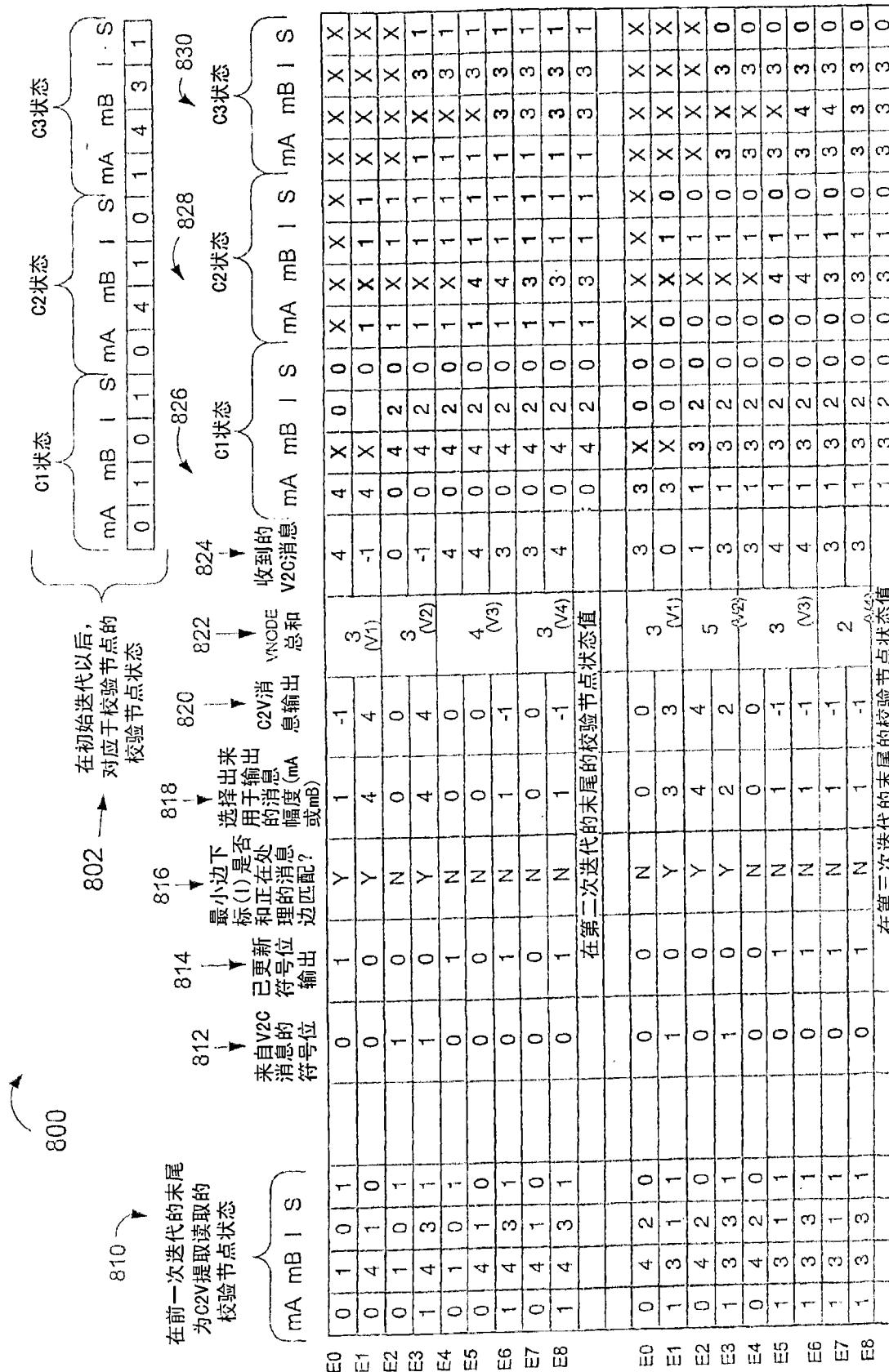


图7





88